3/3/4 (Item 4 from file: 351)
DIALOG(R)File 351:Derwent WPI

013544710 **Image available**
WPI Acc No: 2001-028916/ **200104**
XRPX Acc No: N01-022927

Parallel processing method for detailed analysis of material structure, involves processing elements for every job unit which is formed as combination of loops

Patent Assignee: FUJI XEROX CO LTD (XERF); TAISHO PHARM CO LTD (TAIS); HONDA MOTOR CO LTD (HOND)
Inventor: INABATA S; KITAMURA K; MIYAKAWA N; OBARA S; TAKASHIMA H; YAMADA S
Number of Countries: 002 Number of Patents: 002
Patent Family:

Patent No	Kind	Date	Applicat No	Kind	Date	Week	
JP 2000298658	A	20001024	JP 99105536	A	19990413	200104	B
US 6799151	B1	20040928	US 2000544201	A	20000407	200464	

Priority Applications (No Type Date): JP 99105536 A 19990413
Patent Details:

Patent No	Kind	Lan	Pg	Main IPC	Filing Notes
JP 2000298658	A		35	G06F-015/177	
US 6799151	B1			G06F-017/50	

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-298658
(43)Date of publication of application : 24.10.2000

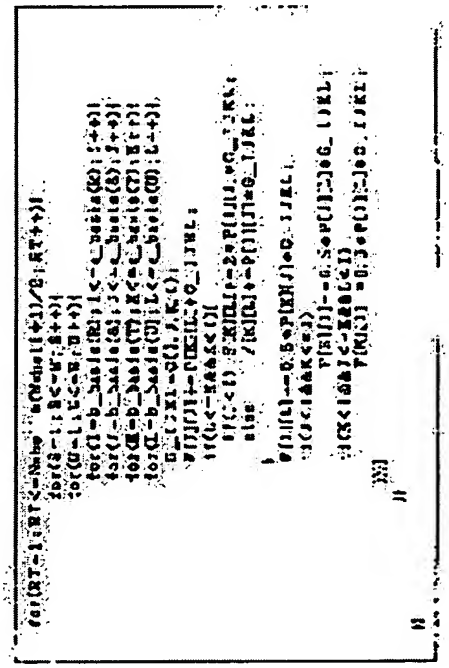
(51)Int.CI. G06F 15/177
G06F 15/16
G06F 17/00

(21)Application number : 11-105536
(22)Date of filing : 13.04.1999

(71)Applicant : FUJI XEROX CO LTD
TAISHO PHARMACEUT CO LTD
(72)Inventor : YAMADA SOU
INAHATA SHINJIROU
MIYAGAWA NOBUAKI
TAKASHIMA HAJIME
KITAMURA KAZUYASU
OBARA SHIGERU

(54) PARALLEL PROCESSING METHOD AND PARALLEL PROCESSOR

(57)Abstract:
PROBLEM TO BE SOLVED: To calculate the matrix element of a large scale at high speed through the use of a parallel processing system having an inexpensive communication means and a small memory by permitting plural processor elements to process jobs for every job unit formed by making second and third loops as one group.
SOLUTION: The outermost loop is defined as the loop with respect to the combination (RT) of contractional shells R and T satisfying the relation of R≤N shell and R ≥T. The second loop is defined as the loop with respect to a contractional shell S and the third loop is defined as the loop with respect to a contractional shell U. In this case, values of S and U may be between '1' and R. A function value G (R, S, T and U) is calculated and a part of a prescribed matrix element F is calculated on the inner side of the third loop. R and T form one job unit with the fixed second and third loops as a group and plural processor elements process jobs for every job unit.



LEGAL STATUS

[Date of request for examination] 17.02.2006

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]
[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報（Ａ）**　(11)特許出願公開番号
特開2000－298658
（P2000－298658A）
(43)公開日　平成12年10月24日(2000.10.24)

(51)Int.Cl.⁷		識別記号	FI		テーマコード*（参考）
Ｇ０６Ｆ	15/177	６８１	Ｇ０６Ｆ 15/177	６８１Ｚ	５Ｂ０４５
	15/16	６１０	15/16	６１０Ｇ	５Ｂ０４９
	17/00		15/20	Ｄ	

審査請求　未請求　請求項の数17　ＯＬ　（全 35 頁）

(21)出願番号　特願平11－105536

(22)出願日　平成11年４月13日(1999.4.13)

(71)出願人　000005496
富士ゼロックス株式会社
東京都港区赤坂二丁目17番22号
(71)出願人　000002819
大正製薬株式会社
東京都豊島区高田３丁目24番１号
(72)発明者　山田　想
神奈川県足柄上郡中井町境430　グリーンテクなかい富士ゼロックス株式会社内
(74)代理人　100091546
弁理士　佐藤　正美

最終頁に続く

(54)【発明の名称】　並列処理方法および並列処理装置

(57)【要約】
【課題】　安価な通信手段と、小容量のメモリを有する多数のプロセッサエレメントとを用いた並列計算によっても、ホスト計算機とプロセッサエレメントとの間の通信性能に律速されることなく、効率的に行列要素計算を行える。
【解決手段】　例えば、分子軌道計算方法において、フォック行列の全ての要素Ｆ（Ｉ，Ｊ）の計算を、最も外側のループは、Ｒ≦Ｎｓｈｅｌｌおよび Ｔ≦Ｒなる関係を満たす縮約シェルＲと縮約シェルＴとの組み合わせ（ＲＴ）に関するループとする。２番目は、縮約シェルＳに関するループ、３番目は縮約シェルＵに関するループとするか、あるいは、２番目は縮約シェルＵに関するループ、３番面は縮約シェルＳに関するループとする。Ｓのとりうる値の範囲を１からＲの間とし、また、Ｕのとりうる値の範囲を１からＲの間とする。３番目のループの内側で、所定の２電子積分の計算およびその結果を用いた所定のフォック行列要素の一部の計算を行うことにより行う。

RT並列アルゴリズム（この発明の実施形態）

```
for(RT=1;RT<=Nshell*(Nshell+1)/2;RT++){
  for(S=1;S<=R;S++){
    for(U=1;U<=R;U++){
      for(I=b_basis(R);I<=e_basis(R);I++){
        for(J=b_basis(S);J<=e_basis(S);J++){
          for(K=b_basis(T);K<=e_basis(T);K++){
            for(L=b_basis(U);L<=e_basis(U);L++){
              G_IJKL=G(I,J,K,L);
              F[I][J]+=P[K][L]*G_IJKL;
              if(L<=K&&K<I){
                if(J<I) F[K][L]+=2*P[I][J]*G_IJKL;
                else    F[K][L]+=P[I][J]*G_IJKL;
              }
              P[I][L]-=0.5*P[K][J]*G_IJKL;
              if(J<I&&K<=J)
                F[K][J]-=0.5*P[I][L]*G_IJKL;
              if(K<I&&J<=K&&L<I)
                F[K][J]-=0.5*P[I][L]*G_IJKL;
            }}}}
  }}}
}
```

1

【特許請求の範囲】

【請求項１】同じ１からＮ（Ｎは正の整数）の範囲にある４つの整数型の変数Ｒ，Ｓ，Ｔ，Ｕを用いて表わされ、Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）＝Ｇ（Ｒ，Ｓ，Ｕ，Ｔ）＝Ｇ（Ｓ，Ｒ，Ｔ，Ｕ）＝Ｇ（Ｓ，Ｒ，Ｕ，Ｔ）＝Ｇ（Ｔ，Ｕ，Ｒ，Ｓ）＝Ｇ（Ｔ，Ｕ，Ｓ，Ｒ）＝Ｇ（Ｕ，Ｔ，Ｒ，Ｓ）＝Ｇ（Ｕ，Ｔ，Ｓ，Ｒ）なる関係を満たす関数Ｇの関数値Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）と；２つの変数Ｔ，Ｕを用いて表わされ、Ｐ（Ｔ，Ｕ）＝Ｐ（Ｕ，Ｔ）なる関係を満たす行列Ｐの要素Ｐ（Ｔ，Ｕ）と；係数Ａ１との積Ａ１・Ｐ（Ｔ，Ｕ）・Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）についての前記範囲の全てのＴおよびＵに関する総和Ｆ１（Ｒ，Ｓ）と、

前記関数値Ｇ（Ｒ，Ｕ，Ｔ，Ｓ）と；前記行列要素Ｐ（Ｔ，Ｕ）と；係数Ａ２との積Ａ２・Ｐ（Ｔ，Ｕ）・Ｇ（Ｒ，Ｕ，Ｔ，Ｓ）に関する前記範囲の全てのＴおよびＵにおける総和Ｆ２（Ｒ，Ｓ）との和Ｆ（Ｒ，Ｓ）＝Ｆ１（Ｒ，Ｓ）＋Ｆ２（Ｒ，Ｓ）を要素とする行列Ｆの全要素の計算を、ホスト計算機と、１つまたは複数個のプロセッサエレメントとを有する並列処理装置を用いて行う並列処理方法において、

前記変数Ｒ，Ｓ，Ｔ，Ｕについて、３重ループを形成し、

前記３重ループの最も外側のループは、Ｒ≦ＮおよびＴ≦Ｒなる関係を満たす変数Ｒと変数Ｔとの組み合わせに関するループとし、

前記最も外側のループの内側の２番目は前記変数Ｓに関するループ、前記２番目よりも内側の３番目は前記変数Ｕに関するループとするか、あるいは前記２番目は変数Ｕに関するループ、前記３番目は変数Ｓに関するループとし、

前記変数Ｓのとりうる値の範囲を１からＲの間とし、

前記変数Ｕのとりうる値の範囲を１からＲの間とし、

前記３番目のループの内側で、所定の前記関数値Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）の計算およびその計算結果を用いた所定の前記行列要素Ｆの一部の計算を行うものであって、

前記２番目および３番目のループをひとまとまりとして１つのジョブ単位を形成し、

前記ジョブ単位毎に前記複数のプロセッサエレメントに処理させることを特徴とする並列処理方法。

【請求項２】Ｎ個（Ｎは正の整数）の縮約シェルを用いて表現される分子のエネルギーを計算する分子軌道計算を、ホスト計算機と、１つまたは複数個のプロセッサエレメントとを有する並列処理装置を用いて行う並列処理方法において、

縮約シェルＲ，Ｓ，Ｔ，Ｕのそれぞれに含まれる原始シェルｒ，ｓ，ｔ，ｕのそれぞれの成分である原始基底関数ｉ，ｊ，ｋ，ｌをインデックスとして用いて表わされる２電子積分関数ｇの関数値ｇ（ｉ，ｊ，ｋ，ｌ）と；

2

前記原始基底関数ｋをひとつの構成要素とする縮約基底関数Ｋおよび前記原始基底関数ｌをひとつの構成要素とする縮約基底関数Ｌとをインデックスとして用いて表わされる密度行列Ｐの要素Ｐ（Ｋ，Ｌ）と；係数Ａ１との積Ａ１・Ｐ（Ｋ，Ｌ）・ｇ（ｉ，ｊ，ｋ，ｌ）の全ての縮約基底関数に関する総和ｆ１（Ｉ，Ｊ）と、

前記２電子積分関数の関数値ｇ（ｉ，ｋ，ｊ，ｌ）と；前記密度行列Ｐの要素Ｐ（Ｋ，Ｌ）と；係数Ａ２との積Ａ２・Ｐ（Ｋ，Ｌ）・ｇ（ｉ，ｋ，ｊ，ｌ）の全ての縮約基底関数に関する総和ｆ２（Ｉ，Ｊ）との和ｆ（Ｉ，Ｊ）＝ｆ１（Ｉ，Ｊ）＋ｆ２（Ｉ，Ｊ）の、前記原始基底関数ｉ，ｊをひとつの構成要素とする前記縮約基底関数Ｉ，Ｊに含まれる全ての前記原始基底関数に関する和で表わされるフォック行列Ｆの全ての行列要素Ｆ（Ｉ，Ｊ）の計算を、

最も外側のループは、Ｒ≦ＮおよびＴ≦Ｒなる関係を満たす前記縮約シェルＲとＴとの組み合わせに関するループとし、

前記最も外側のループの内側の２番目は前記縮約シェルＳに関するループ、前記２番目よりも内側の３番目は前記縮約シェルＵに関するループとするか、あるいは前記２番目は前記縮約シェルＵに関するループ、前記３番目は前記縮約シェルＳに関するループとし、

前記縮約シェルＳのとりうる値の範囲を１からＲの間とし、

前記縮約シェルＵのとりうる値の範囲を１からＲの間とし、

前記３番目のループの内側で、所定の２電子積分の計算およびその結果を用いた所定のフォック行列要素の一部の計算を行うものであって、

前記２番目および３番目のループをひとまとまりとして１つのジョブ単位を形成し、

前記ジョブ単位毎に前記複数のプロセッサ・エレメントに処理を割り当てることを特徴とする並列処理方法。

【請求項３】ホスト計算機と、１つまたは複数個のプロセッサエレメントとを備え、請求項１または請求項２に記載の並列処理方法を実行する並列処理装置であって、

前記ホスト計算機は、少なくとも、

前記複数のプロセッサエレメントに対する前記ＲとＴとが固定されたジョブ単位の割り当て決定と、

前記プロセッサエレメントに対して送信すべき前記行列Ｐの一部の行列要素の選択と、

前記選択された行列要素の前記プロセッサエレメントに対する送信と、

プロセッサ・エレメントから送信された行列Ｆの一部の行列要素の受信と、

前記行列Ｆを用いた前記行列Ｐの更新と、を行い、

前記プロセッサエレメントは、前記ホスト計算機との間でデータの送受信が可能で、少なくとも、

前記ホスト計算機から送信された前記行列Ｐの一部の行

列要素の受信と、
前記Ｓに関するループの制御と、
前記Ｕに関するループの制御と、
前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）または前記関数ｇ（ｉ，ｊ，ｋ，ｌ）の計算と、
前記行列Ｆの一部の行列要素の計算と、
前記行列Ｆの一部の行列要素の前記ホスト計算機に対する送信とを行い、かつ、少なくとも前記ホスト計算機から送信された前記行列Ｐの一部の行列要素と、
前記ホスト計算機へ送信する前記行列Ｆの一部の行列要素とを格納するデータ格納手段を備えることを特徴とする並列処理装置。
【請求項４】請求項１に記載の並列処理方法において、
前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じて行列要素Ｆ（Ｒ，Ｓ）の計算を行うために用いる行列要素Ｐ（Ｔ，Ｕ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じるための行列要素Ｐ（Ｔ，Ｕ）を除いた全てと、
前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じて行列要素Ｆ（Ｔ，Ｕ）の計算を行うために用いる行列要素Ｐ（Ｒ，Ｓ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じるための行列要素Ｐ（Ｒ，Ｓ）を除いた全てと、
前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じて行列要素Ｆ（Ｒ，Ｕ）の計算を行うために用いる行列要素Ｐ（Ｔ，Ｓ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じるための行列要素Ｐ（Ｔ，Ｓ）を除いた全てと、
前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じて行列要素Ｆ（Ｓ，Ｔ）の計算を行うために用いる行列要素Ｐ（Ｒ，Ｕ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じるための行列要素Ｐ（Ｒ，Ｕ）を除いた全てとを含むデータを、前記変数Ｓおよび前記変数Ｕに関するループのうち、外側に位置するものの開始前に、前記ホスト計算機から前記プロセッサエレメントに転送することを特徴とする並列処理方法。
【請求項５】請求項２に記載の並列処理方法において、
前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じて行列要素Ｆ（Ｉ，Ｊ）の計算を行うために用いる行列要素Ｐ（Ｋ，Ｌ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じるための行列要素Ｐ（Ｋ，Ｌ）を除いた全てと、
前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じて行列要素Ｆ（Ｋ，Ｌ）の計算を行うために用いる行列要素Ｐ（Ｉ，Ｊ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じるための行列要素Ｐ（Ｉ，Ｊ）を除いた全てと、
前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じて行列要素Ｆ（Ｉ，Ｌ）の計算を行うために用いる行列要素Ｐ（Ｋ，Ｊ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じるための行列要素Ｐ（Ｋ，Ｊ）を除いた全てと、
前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じて行列要素Ｆ（Ｊ，Ｋ）の計算を行うために用いる行列要素Ｐ（Ｉ，Ｌ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じるための行列要素Ｐ（Ｒ，Ｕ）またはＰ（Ｉ，Ｌ）を除いた全てとを含むデータを、前記インデックスＳおよび前記インデックスＵに関するループのうち、外側に位置するものの開始前に、前記ホスト計算機から前記プロセッサエレメントに転送することを特徴とする並列処理方法。
【請求項６】請求項４に記載の並列処理方法において、
前記ホスト計算機から前記プロセッサエレメントへ送信されるデータが、少なくとも、
前記変数Ｒを表わすデータと前記変数Ｔを表わすデータ、または前記変数ＲとＴとの組み合わせを一意的に表わすデータと、
前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じて行列要素Ｆ（Ｒ，Ｓ）の計算を行うために用いる行列要素Ｐ（Ｔ，Ｕ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じるための行列要素Ｐ（Ｔ，Ｕ）を除いた全ての数値データと、
前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じて行列要素Ｆ（Ｔ，Ｕ）の計算を行うために用いる行列要素Ｐ（Ｒ，Ｓ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じるための行列要素Ｐ（Ｒ，Ｓ）を除いた全ての数値データと、
前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じて行列要素Ｆ（Ｒ，Ｕ）の計算を行うために用いる行列要素Ｐ（Ｔ，Ｓ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じるための行列要素Ｐ（Ｔ，Ｓ）を除いた全ての数値データと、
前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じて行列要素Ｆ（Ｒ，Ｕ）の計算を行うために用いる行列要素Ｐ（Ｔ，Ｓ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）に乗じるための行列要素Ｐ（Ｔ，Ｓ）を除いた全ての数値データと、
前記行列要素Ｐ（Ｔ，Ｕ）およびＰ（Ｒ，Ｕ）を特定す

る前記変数Ｕを表わすデータと、
前記行列要素Ｐ（Ｒ，Ｓ）およびＰ（Ｔ，Ｓ）を特定する変数Ｓを表わすデータと、
上記変数Ｕの個数を表わすデータと、
上記変数Ｓの個数を表わすデータとから構成されることを特徴とする並列処理方法。
【請求項７】請求項５に記載の並列処理方法において、
前記ホスト計算機から前記プロセッサエレメントへ送信されるデータが、少なくとも、
前記インデックスＲを表わすデータと前記インデックスＴを表わすデータ、または前記インデックスＲとＴとの組み合わせを一意的に表わすデータと、
前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じて行列要素Ｆ（Ｉ，Ｊ）の計算を行うために用いる行列要素Ｐ（Ｋ，Ｌ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じるための行列要素Ｐ（Ｋ，Ｌ）を除いた全ての数値データと、
前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じて行列要素Ｆ（Ｋ，Ｌ）の計算を行うために用いる行列要素Ｐ（Ｉ，Ｊ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じるための行列要素Ｐ（Ｉ，Ｊ）を除いた全ての数値データと、
前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じて行列要素Ｆ（Ｉ，Ｌ）の計算を行うために用いる行列要素Ｐ（Ｋ，Ｊ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じるための行列要素Ｐ（Ｋ，Ｊ）を除いた全ての数値データと、
前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じて行列要素Ｆ（Ｉ，Ｌ）の計算を行うために用いる行列要素Ｐ（Ｋ，Ｊ）のうち、前記プロセッサエレメントで計算されないことがあらかじめ判明している、前記関数ｇ（ｉ，ｊ，ｋ，ｌ）に乗じるための行列要素Ｐ（Ｋ，Ｊ）を除いた全ての数値データと、
前記行列要素Ｐ（Ｋ，Ｌ）およびＰ（Ｉ，Ｌ）を特定する、前記インデックスＬで代表される縮約基底関数を構成要素とする前記縮約シェルＵを表わすデータと、
前記行列要素Ｐ（Ｉ，Ｊ）およびＰ（Ｋ，Ｊ）を特定する、前記インデックスＪで代表される縮約基底関数を構成要素とする前記縮約シェルＳを表わすデータと、
前記縮約シェルＵの個数を表わすデータと、
前記縮約シェルＳの個数を表わすデータとから構成されることを特徴とする並列処理方法。
【請求項８】請求項４に記載の並列処理方法において、
前記プロセッサエレメントで計算された行列要素Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）、Ｆ（Ｓ，Ｔ）の全てを、前記Ｓおよび前記Ｕに関するループのうち、外側に位置するものの終了後に、前記プロセッサエレメントから前記ホスト計算機に転送することを特徴とする並列処理方法。

【請求項９】請求項５に記載の並列処理方法において、
前記プロセッサエレメントで計算された行列要素Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）、Ｆ（Ｊ，Ｋ）の全てを、前記Ｓおよび前記Ｕに関するループのうち、外側に位置するものの終了後に、前記プロセッサエレメントから前記ホスト計算機に転送することを特徴とする並列処理方法。
【請求項１０】請求項６または請求項８に記載の並列処理方法において、
前記プロセッサエレメントから前記ホスト計算機へ送信されるデータが、少なくとも、
前記プロセッサエレメントで計算された行列要素Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｓ）から構成されることを特徴とする並列処理方法。
【請求項１１】請求項７または請求項９に記載の並列処理方法において、
前記プロセッサエレメントから前記ホスト計算機へ送信されるデータが、少なくとも、
前記プロセッサエレメントで計算された行列要素Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｊ）から構成されることを特徴とする並列処理方法。
【請求項１２】請求項６に記載の並列処理方法において、
前記プロセッサエレメントにおける前記変数Ｓおよび前記変数Ｕに関するループ制御は、
前記行列要素Ｐ（Ｒ，Ｓ）およびＰ（Ｔ，Ｓ）を特定する変数Ｓを表わすデータを、その最初から前記変数Ｓの個数を表わすデータで示される個数まで順次読み取りながら前記変数Ｓに関するループを制御し、
前記行列要素Ｐ（Ｔ，Ｕ）およびＰ（Ｒ，Ｕ）を特定する変数Ｕを表わすデータを、その最初から前記変数Ｕの個数を表わすデータで示される個数まで順次読み取りながら前記変数Ｕに関するループを制御することにより行われることを特徴とする並列処理方法。
【請求項１３】請求項７に記載の並列処理方法において、
前記プロセッサエレメントにおける前記Ｓおよび前記Ｕに関するループ制御は、
前記行列要素Ｐ（Ｉ，Ｊ）およびＰ（Ｋ，Ｊ）を特定する、インデックスＪで代表される縮約基底関数を構成要素とする縮約シェルＳを表わすデータを、その最初から前記縮約シェルＳの個数を表わすデータで示される個数まで順次読み取りながら前記Ｓに関するループを制御し、
前記行列要素Ｐ（Ｋ，Ｌ）およびＰ（Ｉ，Ｌ）を特定する、インデックスＬで代表される縮約基底関数を構成要素とする縮約シェルＵを表わすデータを、その最初から前記縮約シェルＵの個数を表わすデータで示される個数

まで順次読み取りながら前記Ｕに関するループを制御することにより行われることを特徴とする並列処理方法。

【請求項１４】請求項１０に記載の並列処理方法において、

行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）、Ｐ（Ｔ，Ｓ）と、行列要素Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｓ）との全てを、前記プロセッサエレメントに備えられたデータ格納手段に格納できない場合に、

前記ホスト計算機は、

前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）の全てを前記データ格納手段に格納可能であれば、

前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）の全てとともに前記データ格納手段に前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）、Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）を格納できるように、前記変数Ｓの範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを、同一の前記プロセッサエレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）を前記プロセッサエレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）を送信し、

前記複数の各サブジョブの終了時に、前記行列要素Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）を前記プロセッサエレメントから受信するようにし、

前記行列要素Ｐ（Ｒ，Ｕ）、Ｐ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｕ）の全てを前記データ格納手段に格納することができず、かつ、前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）、Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）の全てを、前記データ格納手段に格納可能であれば、

前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）、Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）の全てとともに前記データ格納手段に行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）を格納できるように、前記変数Ｕの範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを、同一の前記プロセッサエレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）を前記プロセッサエレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）を送信し、

前記複数の各サブジョブの終了時に、前記行列要素Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）を前記プロセッサエレメントから受信することを特徴とする並列処理方法。

【請求項１５】請求項１１に記載の並列処理方法において、

行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）、Ｐ（Ｋ，Ｊ）と、行列要素Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｊ）の全てをプロセッサエレメントに備えられたデータ格納手段に格納できない場合に、

前記ホスト計算機は、

前記行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）の全てを前記データ格納手段に格納可能であれば、

前記行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）の全てとともに前記データ格納手段に前記行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）、Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）を格納できるように、前記縮約シェルＳの範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを同一の前記プロセッサエレメントに割り当て、

前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）を送信し、

前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）を前記プロセッサエレメントから受信し、

前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）を送信し、

前記複数の各サブジョブの終了時に、前記行列要素Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）を前記プロセッサエレメントから受信するようにし、

前記行列要素Ｐ（Ｉ，Ｌ）、Ｐ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｌ）の全てを前記データ格納手段に格納することができず、かつ、前記行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）、Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）の全てを前記データ格納手段に格納可能であれば、

前記行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）、Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）の全てとともに前記データ格納手段に行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）を格納できるように、前記縮約シェルＵの範囲を区切って、前記ジョブを複数のサブジョブに分割し、

前記分割された複数のサブジョブを、同一の前記プロセッサエレメントに割り当て、
前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）を送信し、
前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）を前記プロセッサエレメントから受信し、
前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）を送信し、
前記複数の各サブジョブの終了時に、前記行列要素Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）を前記プロセッサエレメントから受信することを特徴とする並列処理方法。
【請求項１６】請求項１０に記載の並列処理方法において、
行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）、Ｐ（Ｔ，Ｓ）と、行列要素Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｓ）の全てを、前記プロセッサエレメントに備えられたデータ格納手段に格納できない場合に、
前記ホスト計算機は、
前記行列要素Ｐ（Ｒ，Ｓ）と前記行列要素Ｐ（Ｔ，Ｓ）の個数の和が、前記行列要素Ｐ（Ｒ，Ｕ）と前記行列要素Ｐ（Ｔ，Ｕ）の個数の和よりも大きく、かつ、前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）の全てを、前記データ格納手段に格納可能でなければ、
前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）が前記データ格納手段に格納可能となるように、前記変数Ｕの範囲を等分割し、
前記等分割された前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）とともに前記データ格納手段に前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）、Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）を格納できるように、前記変数Ｓの範囲を区切って、前記ジョブを複数のサブジョブに分割し、
前記分割された複数のサブジョブを、同一の前記プロセッサエレメントに割り当て、
前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）を送信し、
前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）を前記プロセッサエレメントから受信し、
前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）を送信し、
前記複数の各サブジョブの終了時に、前記行列要素Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）を前記プロセッサエレメントから受信するようにし、
前記行列要素Ｐ（Ｒ，Ｓ）と前記行列要素Ｐ（Ｔ，Ｓ）の個数の和が、前記行列要素Ｐ（Ｒ，Ｕ）と前記行列要素Ｐ（Ｔ，Ｕ）の個数の和よりも小さいか、または等しく、かつ、前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）、Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）の全てを前記データ格納手段に格納可能でなければ、
前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）、Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）が前記データ格納手段に格納可能となるように、前記変数Ｓの範囲を等分割し、
前記等分割された前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）、Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）とともに前記データ格納手段に前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）、Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）を格納できるように、前記変数Ｕの範囲を区切って、前記ジョブを複数のサブジョブに分割し、
前記分割された複数のサブジョブを、同一の前記プロセッサ・エレメントに割り当て、
前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｒ，Ｓ）、Ｐ（Ｔ，Ｓ）を送信し、
前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素Ｆ（Ｒ，Ｓ）、Ｆ（Ｔ，Ｓ）を前記プロセッサ・エレメントから受信し、
前記複数の各サブジョブの開始時に、前記プロセッサ・エレメントに対して前記行列要素Ｐ（Ｔ，Ｕ）、Ｐ（Ｒ，Ｕ）を送信し、
前記複数の各サブジョブの終了時に、前記行列要素Ｆ（Ｔ，Ｕ）、Ｆ（Ｒ，Ｕ）を前記プロセッサエレメントから受信する、
ことを特徴とする並列処理方法。
【請求項１７】請求項１１に記載の並列処理方法において、
行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）、Ｐ（Ｋ，Ｊ）と、行列要素Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｊ）の全てを、前記プロセッサエレメントに備えられたデータ格納手段に格納できない場合に、
前記ホスト計算機は、
前記行列要素Ｐ（Ｉ，Ｊ）と前記行列要素Ｐ（Ｋ，Ｊ）の個数の和が、前記行列要素Ｐ（Ｉ，Ｌ）と前記行列要素Ｐ（Ｋ，Ｌ）の個数の和よりも大きく、かつ、前記行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）の全てを前記データ格納手段に格納可能でなければ、
行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）を前記データ格納手段に格納可能となるように縮約シェルＵの範囲を等分割し、
前記等分割された行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）とともに前記データ

格納手段に行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）、Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）を格納できるように縮約シェルＳの範囲を区切ってジョブを複数のサブジョブに分割し、
前記のように区切られた複数のサブジョブを同一の前記プロセッサエレメントに割り当て、
前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ前記プロセッサ・エレメントに対して行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）を送信し、
前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ行列要素Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）を前記プロセッサ・エレメントから受信し、
前記複数の各サブジョブの開始時に前記プロセッサエレメントに対して行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）を送信し、
前記複数の各サブジョブの終了時に行列要素Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）を前記プロセッサエレメントから受信するようにし、
行列要素Ｐ（Ｉ，Ｊ）と前記行列要素Ｐ（Ｋ，Ｊ）の個数の和が行列要素Ｐ（Ｉ，Ｌ）と前記行列要素Ｐ（Ｋ，Ｌ）の個数の和よりも小さいか、または等しく、かつ、前記行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）、Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）の全てを前記データ格納手段に格納可能でなければ、
前記行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）、Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）を前記データ格納手段に格納可能となるように、前記縮約シェルＳの範囲を等分割し、
前記等分割された前記行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）、Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）とともに前記データ格納手段に前記行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）、Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）を格納できるように、前記縮約シェルＵの範囲を区切って、前記ジョブを複数のサブジョブに分割し、
前記分割された複数のサブジョブを、同一の前記プロセッサエレメントに割り当て、
前記複数のサブジョブのうちの最初のサブジョブの開始時にだけ、前記プロセッサエレメントに対して前記行列要素Ｐ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｊ）を送信し、
前記複数のサブジョブのうちの最後のサブジョブの終了時にだけ、前記行列要素Ｆ（Ｉ，Ｊ）、Ｆ（Ｋ，Ｊ）を前記プロセッサエレメントから受信し、
前記複数の各サブジョブの開始時に、前記プロセッサエレメントに対して行列要素Ｐ（Ｋ，Ｌ）、Ｐ（Ｉ，Ｌ）を送信し、
前記複数の各サブジョブの終了時に、前記行列要素Ｆ（Ｋ，Ｌ）、Ｆ（Ｉ，Ｌ）を前記プロセッサエレメントから受信することを特徴とする並列処理方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】この発明は、大規模で、特定の対称性を有する行列要素計算、特に、非経験的分子軌道法を用いた分子シミュレーションにおいてフォック行列要素計算を高速に処理するために用いて好適な、並列処理方法および並列処理装置に関する。

【０００２】

【従来の技術】化学の分野において、分子の状態や挙動を数値的に解析する手法として、分子軌道法、分子動力学法、モンテカルロ法などがある。その中でも、非経験的分子軌道計算法は、第一原理に基づいた量子力学的計算で、分子中の電子の挙動を記述することを目的としている。そのため、この手法は、分子シミュレーションの基盤として位置づけられ、物質構造や化学反応の詳細な解析に用いられている工業的に重要な手法である。

【０００３】非経験的分子軌道計算法では、分子を構成する原子の原子核と軌道電子との距離の２乗に経験的な定数を乗じたものを指数とする指数関数の逆数あるいはそれらの線形結合を基底関数とし、この基底関数を１つの原子に対して複数個用意する。これらの基底関数の線形結合で、分子内の軌道電子の波動関数、すなわち分子軌道を記述する。

【０００４】分子軌道における基底関数の線形結合係数を決めることが、非経験的分子軌道計算法での主要な処理であるが、その計算には、基底関数の数の４乗に比例した計算量と記憶容量を必要とする。そのため、非経験的分子軌道計算法は、現状では、１００原子程度の規模の分子系に適用されているに過ぎない。生命現象／化学現象の分子論的解明を、より現実的なものとするためには、数１０００原子規模の分子系への適用も視野に入れた、非経験的分子軌道計算専用の計算システムの開発が必須である。

【０００５】［非経験的分子起動法の概要］非経験的分子軌道計算法では、分子の状態Ψを、分子中の電子の空間的な軌道に相当する電子軌道関数ϕ_μを用いて記述する。ここでμは複数ある分子軌道のμ番目という意味の添え字である。分子軌道ϕ_μは、原子軌道χ_Iの線形結合で、図２３の（数式１）のように近似的に表わされる。

【０００６】ここで、（数式１）において、Ｉは複数ある原子軌道のＩ番目という意味の添え字である。なお、原子軌道は基底関数とも呼ばれることがある。この明細書中では、以降、原子軌道のことを基底関数と呼ぶ。また、数式１に現れる$C_{I\mu}$は、線形結合係数である。数式１におけるＩに関する総和は、計算の対象とする分子を構成する全ての基底関数に関するものである。

【０００７】さて、量子力学的に分子軌道を記述するためには、良く知られるパウリの排他律を、分子内の電子の状態が満たさなければならない。電子のスピンを考慮に入れて、パウリの排他律を満たすように、２ｎ電子系の分子の状態Ψを記述する表式として、図２３の（数式２）のようなスレーター行列式が用いられる。ここで、

（数式２）において、α（ｘ）およびはβ（ｘ）、ｘ番目の電子のスピンが、それぞれ、上向きおよび下向きの状態を表わしている。

【０００８】２ｎ電子系に対するハミルトニアンＨは、１電子部分H_1と２電子部分H_2との和という形式で、図２３の（数式３）から（数式５）のように書き表される。

【０００９】図２３の（数式４）において、右辺の（・・・）内の第１項は、電子ｐの運動エネルギー、第２項はｐ番目の電子とＡ番目の原子核との相互作用である。（数式４）において、Σ_p（この明細書でΣ_iは、ｉについての総和を取ることを表すものとする。以下、同じ）は全電子に関する総和、Σ_Aは全原子核に関する総和、Z_Aは原子核Ａの電荷、r_{pA}は電子ｐと原子核Ａとの距離である。

【００１０】また、（数式５）は、電子ｐと電子ｑとの間の相互作用を表わしており、$\Sigma_p\Sigma_{q(>p)}$は２個の電子の組み合わせに関する総和、r_{pq}は電子ｐ，ｑ間の距離である。

【００１１】上記のハミルトニアンＨと、（数式２）のスレーター行列式とを用いると、分子エネルギーの期待値εが、図２４の（数式６）～（数式９）のように表わされる。

【００１２】（数式６）において、Σ_μおよびΣ_νは、ｎ個（ｎは正の整数）ある分子軌道に関する総和である。（数式７）は「コア積分」と呼ばれ、代表として番号１の電子について書かれている。また、（数式８）および（数式９）は、それぞれ「クーロン積分」および「交換積分」と呼ばれ、代表として電子１および電子２について書かれている。

【００１３】（数式６）を基底関数を用いて書き直すと、図２５に示す（数式１０）～（数式１３）に示すようなものになる。（数式１３）で表わされる積分を、「２電子間反発積分」あるいは省略して「２電子積分」と呼ぶ。

【００１４】（数式１０）で表わされる分子エネルギーの期待値εは、$C_{I\mu}$という未知数を含んでおり、このままでは数値が得られない。$C_{I\mu}$は、（数式１）における線形結合定数であり、μは１からｎ（分子軌道の数）の整数、Ｉは１からＮ（Ｎが基底関数の数であり、正の整数）の整数である。以下では、$C_{I\mu}$を要素とするＮ×ｎ行列Ｃを係数行列と呼ぶ。

【００１５】期待値εが最小となるように係数行列を決定し、基底状態の波動関数Ψを求める手法の１つとして、ハートリー・フォック・ローサーンの変分法（以下、ＨＦＲ法と略称する）が用いられる。導出過程は省略し、ＨＦＲ法の結果として得られる式を、図２６の（数式１４）～（数式１８）に示す。

【００１６】F_{IJ}はフォック行列要素、P_{KL}は密度行列要素と、それぞれ呼ばれる。以下の説明では、これらをＦ（Ｉ，Ｊ）、Ｐ（Ｋ，Ｌ）のように表記する場合がある。これらは、１からＮの値をとる各Ｉ，Ｊ，Ｋ，Ｌに対して数値を持っており、それぞれＮ×Ｎ行列の形で表わされる。

【００１７】（数式１４）を解くことにより、係数行列が求まる。（数式１４）は、１からｎの間の全てのμ、および１からＮの間の全てのＩに対して存在するので、ｎ×Ｎ本の連立方程式になっている。

【００１８】（数式１４）を解いて得られた係数行列Ｃの計算には、密度行列Ｐが用いられている。密度行列Ｐは、（数式１８）に示すように係数行列Ｃから計算される。そのため、具体的な計算手順としては、まず、適当に係数行列Ｃを与えておき、それを用いて計算した密度行列Ｐを使って、（数式１５）でフォック行列Ｆを計算し、（数式１４）の連立方程式を解いて新たな係数行列Ｃを得る。密度行列Ｐの元となるＣと、結果として得られるＣとの間の差が十分小さく、すなわち自己無撞着になるまで、上記の計算を繰り返し行う。この反復計算を自己無撞着計算（以下、ＳＣＦ計算と称する）と呼ぶ。

【００１９】実際の計算で最も時間を要するのは、（数式１５）のフォック行列要素F_{IJ}の計算である。これは、全てのＩ，Ｊに対して、この（数式１５）を計算しなければならないこと、および各Ｉ，Ｊの組み合わせに対して、密度行列要素P_{KL}のＫ，Ｌに関する和を計算しなければならないことに起因する。

【００２０】ＳＣＦ計算の手法には２通りある。１つはディスクストレージＳＣＦ法と呼ばれる手法で、１回目のＳＣＦ計算の際に得た２電子積分の値を全てディスクに保存しておき、２回目以降は必要な２電子積分をディスクから取り出して用いる手法である。もう１つはダイレクトＳＣＦ法と呼ばれる手法で、ＳＣＦ計算の度に２電子積分の計算をやり直す手法である。

【００２１】現在では、ディスク容量の制限やアクセス時間の大きさなどから、後者のダイレクトＳＣＦ法を用いるのが主流である。このダイレクトＳＣＦ法による分子軌道計算においては、ＳＣＦ計算の１回あたりに、N^4にほぼ比例する個数の２電子積分の計算を行わなければならないため、２電子積分計算を高速に行うことが分子軌道計算を高速化することに直結する。

【００２２】２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）、密度行列Ｐ（Ｋ，Ｌ）、およびフォック行列Ｆ（Ｉ，Ｊ）の対称性に関して、ここで言及しておく。

【００２３】２電子積分は、（数式１３）から明らかなように、図２６の（数式１９）に示すような対称性を有している。したがって、（数式１９）の内の１つに関して数値を得ることができれば、他の７つについても数値が得られたことになる。

【００２４】また、図２６の（数式１８）から、

Ｐ（Ｋ，Ｌ）＝Ｐ（Ｌ，Ｋ）

であることがわかり、図２６の（数式１５）および図２

５の（数式１１）から、
Ｆ（Ｉ，Ｊ）＝Ｆ（Ｊ，Ｉ）
であることがわかる。

【００２５】［縮約基底関数と原始基底関数］非経験的分子軌道法では、図２７の（数式２０）に示すような基底関数が一般的に用いられる。この（数式２０）において、ｒ，ｎ，Ｒはベクトルであり、添え字ｘ，ｙ，ｚの付いたものがその成分である。ｒは電子の座標、ｎは電子の角運動量、Ｒは原子核の座標である。

【００２６】$n_x+n_y+n_z=\lambda$は、角運動量の大きさであり、軌道量子数とも呼ばれる。この軌道量子数λが、０の場合にその軌道をｓ軌道、１の場合にその軌道をｐ軌道、２の場合にその軌道をｄ軌道などと呼ぶ。

【００２７】ζ_mは軌道指数であり、軌道の空間的な広がり具合を示す。軌道指数の異なる複数の軌道の線形結合で１つの基底関数を表わす場合があり、そのようにして表わした基底関数を縮約基底関数と呼び、線形結合係数d_mを縮約係数と呼ぶ。これに対して、線形結合される前の、図２７の（数式２１）の形の関数ψを原始基底関数と呼ぶ。

【００２８】縮約基底関数χは、Ｉ，Ｊ，Ｋ，Ｌのように大文字で番号付けをし、また、原始基底関数ψは、ｉ，ｊ，ｋ，ｌのように小文字で番号付けするのが慣例であり、本明細書中でもこれに従う。

【００２９】［縮約シェルと原始シェル］軌道量子数が１の場合の縮約基底関数には、ｎ＝（１，０，０）の場合、ｎ＝（０，１，０）の場合、ｎ＝（０，０，１）の場合の３通りが存在する。同様に、軌道量子数が２の場合には６通り（あるいは、基底関数の構成の仕方によっては５通り）の縮約基底関数が存在する。

【００３０】（数式２０）のうちの図２７の（数式２２）で示す部分が共通な、これら複数の縮約基底関数の集合を、縮約シェルと呼ぶ。ｐ軌道の縮約シェルは３つの縮約基底関数で構成され、また、ｄ軌道の縮約シェルは６つ（または５つ）の縮約基底関数で構成される。ｓ軌道の場合にも、便宜上１つの縮約基底関数の集合を縮約シェルと呼ぶ。

【００３１】（数式２１）のうちのｅｘｐ［－ζ（ｒ－Ｒ）^{2}］の部分が共通な、原始基底関数の集合を、同様に原始シェルと呼ぶ。縮約シェルは、Ｒ，Ｓ，Ｔ，Ｕのように大文字で番号付けをし、原始シェルは、ｒ，ｓ，ｔ，ｕのように小文字で番号付けするのが慣例であり、本明細書中でもこれに従う。

【００３２】分子軌道計算の実施に際しては、計算の対象とする分子を構成する原子毎に軌道量子数の異なる複数の縮約シェルを用意し、それら全ての集合を基底関数のセットとして用いる。原子核座標Ｒと軌道量子数λとの組み合わせ（Ｒ，λ）で、１つの縮約シェルを表わすことができる。

【００３３】［２電子積分の表式］縮約基底関数で表わされる２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）は、原始基底関数を用いると、図２７の（数式２３）のように表わされる。ここで、ｇ（ｉ，ｊ，ｋ，ｌ）は、図２７の（数式２４）のように表すことができる。

【００３４】Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）を、縮約基底関数で表現した２電子積分と呼び、ｇ（ｉ，ｊ，ｋ，ｌ）を、原始基底関数で表現した２電子積分と呼ぶが、以降の説明では、どちらも単に２電子積分と呼ぶ場合がある。ｇ（ｉ，ｊ，ｋ，ｌ）も、図２７の（数式２５）で示すような対称性を有している。

【００３５】さて、原始基底関数ψは、その角運動量ｎ、軌道指数ζ、原子核座標Ｒの組み合わせで、一意的に示すことができる。ｉ，ｊ，ｋ，ｌ番目の原始基底関数が、図１９に示す表１のような角運動量、軌道指数、原子核座標を有するものと仮定する。

【００３６】説明の煩雑さを避けるために、以下の説明では、原始基底関数の番号ｉ，ｊ，ｋ，ｌの代わりに、それぞれの角運動量ａ，ｂ，ｃ，ｄを用いて２電子積分を［ａｂ，ｃｄ］のように表わすことにする。

【００３７】上記のように用意された基底関数セットを用いて２電子積分を計算する効率的な手法を、文献１（Ｓ．Ｏｂａｒａ　ａｎｄ　Ａ．Ｓａｉｋａ，ＪＣＰ，ｖｏｌ．８４，ｎｏ．７，ｐ．３９６４，１９８６）に従って説明する。

【００３８】まず、ａ，ｂ，ｃ，ｄが全てｓ軌道、すなわち$a=0_a=(0,0,0)$，$b=0_b=(0,0,0)$，$c=0_c=(0,0,0)$，$d=0_d=(0,0,0)$である場合には、（数式２４）の２電子積分は、図２８の（数式２６）～（数式３４）に示すように求まる。

【００３９】ここで、（数式２６）に現れる$[\cdot\cdot,\cdot\cdot]^{(m)}$は補助積分、ｍは補助インデックスであるが、これらについては後で述べる。（数式２７）の積分範囲は、０から１である。

【００４０】また、ａ，ｂ，ｃ，ｄのうち１つでもｓ軌道以外のものがある場合には、図２９の（数式３５）および（数式３６）に示す漸化式を用いて計算する。

【００４１】（数式３５）で、添え字のｉは、ｘまたはｙまたはｚ成分であることを示す。また、1_iは、ｉ成分のみ１で、他は０であるようなベクトルである。さらに、$N_i(n)$は、角運動量ｎのｉ成分の値を示すものである。（数式３５）は、左辺の補助積分に現れる角運動量の１つは右辺では確実に１以上減少し、また、左辺の補助積分の補助インデックスは右辺では同じあるいは１だけ増加する、という性質を有している。

【００４２】補助積分$[\cdot\cdot,\cdot\cdot]^{(m)}$は、補助インデックスｍが０であるときに、２電子積分［・・，・・］に正確に一致するものであり、２電子積分の計算を補助するものである。どんなに角運動量の大きな基底関数を含んだ２電子積分であっても、（数式３５）を繰り

返し用いて角運動量を減少させ、最後には、全て角運動量が（０，０，０）であるような補助積分に行き着くことができる。角運動量が（０，０，０）であるような補助積分は、（数式２６）を用いて計算できるので、その数値と適当な係数を乗じ、加算すれば２電子積分の数値が得られる。

【００４３】実際の計算は、以下のように行う。まず、（数式３５）に従って、２電子積分を８つ以下の補助積分を用いた形式に表わす。ここで現れた補助積分に対して、さらに、（数式３５）を適用する。このような手続きを繰り返して、全て角運動量が（０，０，０）であるような補助積分に行き着くまでの道筋を、計算手順として記録しておく。

【００４４】次に、（数式２６）を用いて角運動量が（０，０，０）であるような補助積分を計算し、そこから出発して、先ほどの計算手順をたどりながら補助積分の数値を計算していき、最後に目的とする２電子積分の数値を得る。

【００４５】（数式３５）が有するもう一つの重要な性質は、２電子積分に現れる４つの角運動量の組み合わせが同じであれば、軌道指数や原子核座標の組み合わせが異なっていても、上記の計算手順としては全く同じものを用いることができることである。計算の実行に際しては、軌道指数や原子核座標に応じて補助積分に乗じる係数を変えてやるだけで良い。

【００４６】［カットオフ］上述したように、計算しなければならない縮約基底関数で表わした２電子積分の数は、縮約基底関数の数Ｎに対してN^4となる。実際に数値を得なければならないのは、原始基底関数で表わした２電子積分の方であるが、その総数は、縮約基底関数で表わした２電子積分の数の数倍から数１０倍（縮約基底関数を構成する原始基底関数の数、すなわち縮約数に依存する）に及ぶ。

【００４７】この個数を減らす手法として、第１に考えられるのは、（数式１９）あるいは（数式２５）に記した対称性を利用することである。しかしながら、この方法では最も効率化を行っても２電子積分の計算量は１／８にしかならない。

【００４８】もう１つの手法は、計算精度の観点から、不必要と判断できる２電子積分の計算を、積極的に排除する方法である。不必要な２電子積分の判断は、以下のように行うことができる。

【００４９】上述したように、全ての２電子積分の数値は、（数式２６）に示した全て角運動量が（０，０，０）であるような補助積分［００，００］$^{(m)}$の数値に基づいて計算される。したがって、２電子積分の数値の、フォック行列要素の数値への寄与が計算誤差の程度であるかどうかを、［００，００］$^{(m)}$の数値で判断することが可能である。さらに、［００，００］$^{(m)}$の数値の大きさは、（数式２９）に示した関数Ｋ（ζ，ζ’，Ｒ，Ｒ’）の値から、さらに、それは、図２９の（数式３７）の大きさから判断することができる。

【００５０】したがって、（ζ_a，Ａ，ζ_b，Ｂ）の数値の組み合わせで、（数式２５）の１つ目の関数Ｋの大きさを見積ることで、２電子積分［ａｂ，＊＊］を計算する必要があるかどうかを判断し、また、（ζ_c，Ｃ，ζ_d，Ｄ）の数値の組み合わせで、（数式２６）の２つ目の関数Ｋの大きさを見積ることで、２電子積分［＊＊，ｃｄ］を計算する必要があるかどうかを判断することができる。

【００５１】このようにして、不必要な２電子積分の計算を排除することを、「カットオフする」と呼ぶことにする。上記の例で、ａおよびｂの情報だけから判断してカットオフする場合には、ａｂでのカットオフ、ｃおよびｄの情報だけから判断してカットオフする場合には、ｃｄでのカットオフ、と呼ぶ場合がある。このように、ａｂだけで、あるいはｃｄだけでカットオフするかどうかの判断ができるのは、図２９の（数式３７）の最大値が１で下限値が０だからである。このようにカットオフを行うことにより、計算しなければならない２電子積分は、概略でN^2に比例する個数となり、計算量を大幅に低減できる。

【００５２】上述のことから、Ｎが大きい場合には、２電子積分の対称性を利用することによる効果よりも、カットオフによる計算量低減の効果の方が桁違いに大きく、これを取り入れることによって、非経験的分子軌道計算におけるフォック行列の計算に要する処理時間が大きく短縮できることがわかる。

【００５３】［分子軌道計算機システムの例］フォック行列要素の計算を、並列計算機を用いて高速に行うシステムの例として、文献２（白川他，”超高速分子軌道計算専用機ＭＯＥのアーキテクチャ”，電子情報通信学会技術報告，ｖｏｌ．ＣＰＳＹ９６－４６，ｎｏ．５，ｐｐ．４５－５０，１９９６）に記載のシステムがある。

【００５４】この文献２では、ホスト計算機に、複数個のプロセッサエレメントをバスを介して接続して並列処理を行うシステムが示されている。この文献２では、このような構成を有する並列処理システムのアーキテクチャの検討に際して、Ｒ，Ｓ，Ｔ，Ｕの４つのインデックスで構成される４重ループのまわし方および並列化を行う部分の種々の方法に関して、全体の計算量およびプロセッサエレメントに必要なメモリ量を見積っている。

【００５５】文献２に記載されている並列処理システムは、個々のプロセッサ・エレメントが高い演算処理能力を有する上、システム全体を低価格で実現することが可能であるため、コストパフォーマンスの優れた計算システムを提供することができる。しかしながら、文献２では、前述したカットオフを考慮する場合の方法や具体的なループの制御方法への言及がなく、効率的な処理が行えるかどうかが不明であった。

【００５６】［Ｉ．Ｆｏｓｔｅｒらの方法］フォック行列要素の計算を、並列計算機を用いて効率的に行うアルゴリズムとして、文献３（Ｉ．Ｔ．Ｆｏｓｔｅｒ，ｅｔ．ａｌ．，”Ｔｏｗａｒｄ Ｈｉｇｈ－Ｐｅｒｆｏｒｍａｎｃｅ Ｃｏｍｐｕｔａｔｉｏｎａｌ Ｃｈｅｍｉｓｔｒｙ：Ｉ．Ｓｃａｌａｂｌｅ Ｆｏｃｋ Ｍａｔｒｉｘ Ｃｏｎｓｔｒｕｃｔｉｏｎ Ａｌｇｏｒｉｔｈｍｓ”，Ｊｏｕｒｎａｌ ｏｆ Ｃｏｍｐｕｔａｔｉｏｎａｌ Ｃｈｅｍｉｓｔｒｙ，ｖｏｌ．１７，ｎｏ．１，ｐ．１０９，１９９６）に記載のアルゴリズムがある。

【００５７】この文献３では、幾つかのフォック行列要素計算アルゴリズムに関して、その計算量およびホスト計算機と、複数のプロセッサ・エレメントとの間の通信量を解析している。その内容を、以下に説明する。

【００５８】第１のアルゴリズムは、カノニカル法と呼ばれる最も簡単なアルゴリズムである。この手法では、１つのプロセッサ・エレメントに、図３０に示す（数式３８）の関係を満たす４個の縮約基底関数Ｉ，Ｊ，Ｋ，Ｌと、６個の密度行列要素P_{IJ}，P_{IK}，P_{IL}，P_{JK}，P_{JL}，P_{KL}とを渡し、プロセッサエレメントには２電子積分を計算させて、フォック行列要素F_{IJ}，F_{IK}，F_{IL}，F_{JK}，F_{JL}，F_{KL}の一部を、図３０の（数式３９）に従って計算させる。

【００５９】１つの２電子積分を計算する間に、ホスト計算機とプロセッサエレメントとの間で通信される行列要素の数を、ｐｅｒＥＲＩという単位で勘定することにすると、この場合には、通信データ数が１２［ｐｅｒＥＲＩ］となる。

【００６０】第２のアルゴリズムは、トリプルソート法と呼ばれるアルゴリズムである。図３１の（数式４０）の関係を満たす４個の縮約基底関数Ｉ，Ｊ，Ｋ，Ｌと、６個の密度行列要素P_{IJ}，P_{IK}，P_{IL}，P_{JK}，P_{JL}，P_{KL}とを、１つのプロセッサエレメントに渡し、そのプロセッサエレメントに３つの２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ），Ｇ（Ｉ，Ｋ，Ｊ，Ｌ），Ｇ（Ｉ，Ｌ，Ｊ，Ｋ）を計算させて、フォック行列要素F_{IJ}，F_{IK}，F_{IL}，F_{JK}，F_{JL}，F_{KL}の一部を、図３１の（数式４１）に従って計算させる。

【００６１】この場合、３つの２電子積分を計算する間に、６つの密度行列要素と、６つのフォック行列要素の転送が必要であるので、通信データ数は、４［ｐｅｒＥＲＩ］である。したがって、通信データ数の観点からカノニカル法より優れていると言える。

【００６２】しかしながら、プロセッサエレメントにおける原始基底関数で表わした２電子積分の計算時間を、例えば2マイクロ秒（＝10^{-6}秒、以下ではμｓと記す）と仮定し、また、縮約基底関数の平均の縮約数を１．５と仮定し、密度行列要素やフォック行列要素が、倍精度の浮動小数点数すなわち６４ビットのデータサイズであると仮定すると、１つの縮約基底関数で表わした２電子積分の計算時間は、約１０μｓとなり、１つのプロセッサエレメント当たりで、２５．６Ｍｂｐｓ（２５．６×10^6 ｂｉｔ ｐｅｒ ｓｅｃｏｎｄ）の通信性能が、ホスト計算機とプロセッサエレメントとの間で必要とされることになる。

【００６３】計算性能を上げるために、プロセッサエレメント数Ｍを、例えば１００とした場合には、全体の通信性能としては、２５６０Ｍｂｐｓが要求されることとなるが、現在の技術で、このような通信性能を達成するのは容易でない。

【００６４】安価な通信手段、例えばＩＥＥＥ１３９４バス規格で定められたシリアル通信では、２００Ｍｂｐｓの通信性能が実現できるが、それを用いてトリプルソート法を採用したフォック行列要素並列計算を行うと、全体の処理時間が通信時間で律速されてしまい、行列の対称性を利用した計算時間低減の効果は得られなくなってしまう。

【００６５】第３のアルゴリズムは、単純ブロック法と呼ばれる手法である。これは、さらに、カノニカル法に基づいたものと、トリプルソート法に基づいたものとに細分類できる。この第３のアルゴリズムは、縮約基底関数をブロック化しておくことにより、密度行列要素やフォック行列要素の利用効率を高め、通信量を低減する手法である。

【００６６】トリプルソート法に基づいて、その手法を説明する。まず、Ｎ個ある縮約基底関数を、I_C 個毎に、ｎ（＝Ｎ÷I_C）個のブロックに分割する。ブロックの番号を、I_B，J_B，K_B，L_Bのように表わすことにする。次に、図３１の（数式４２）の関係を満たす４個の縮約基底関数ブロックI_B，J_B，K_B，L_Bと、６個の密度行列要素ブロックＰ（I_B，J_B），Ｐ（I_B，K_B），Ｐ（I_B，L_B），Ｐ（J_B，K_B），Ｐ（J_B，L_B），Ｐ（K_B，L_B）を、１つのプロセッサエレメントに渡す。渡される密度行列要素の数は$6I_C^2$個となる。

【００６７】プロセッサエレメントが計算する２電子積分は、Ｇ（Ｉ，Ｊ，Ｋ，Ｌ），Ｇ（Ｉ，Ｋ，Ｊ，Ｌ），Ｇ（Ｉ，Ｌ，Ｊ，Ｋ）に対応する$3I_C^4$個の２電子積分であり、プロセッサエレメントは、前述の（数式４１）と同様に、フォック行列要素ブロックＦ（I_B，J_B），Ｆ（I_B，K_B），Ｆ（I_B，L_B），Ｆ（J_B，K_B），Ｆ（J_B，L_B），Ｆ（K_B，L_B）を計算して、ホスト計算機に送り返す。

【００６８】このときに送り返されるフォック行列要素数も、$6I_C^2$個である。その結果、通信データ数は、$12I_C^2 \div 3I_C^4 = 4/I_C^2$［ｐｅｒＥＲＩ］となる。つまり、ブロック内の縮約基底関数の数を多くすればするほど、密度行列要素やフォック行列要素の利用効率が高まり、通信量が低減する。なお、カノニカル法

の場合には、図３１の（数式４２）に代わり（数式４３）を用いる。

【００６９】さらに、通信量を低減する第４の手法として、列ブロック法がある。これは、単純ブロック法において、I_B，J_B，K_B の組み合わせが同じで、L_B だけが異なる計算を、全て１つのプロセッサエレメントに割り当てる手法である。この手法のフローチャートを図２０に示した。なお、図２０において破線で示した矢印の部分は、その後の処理がその前の処理で律速されるのでなく、他の処理系からの情報の入力待ちとなることを示している。

【００７０】列ブロック法も、さらに、カノニカル法に基づいたものとトリプルソート法に基づいたものとに細分類できるが、トリプルソート法に基づいて、図２０のフローチャートを参照しながら説明する。

【００７１】まず、最初に単純ブロック法と同様に、Ｎ個ある縮約基底関数を、I_C 個毎にｎ（＝Ｎ÷I_C）個のブロックに分割する（ステップＳ１）。次に、ホスト計算機は、特定のプロセッサエレメントに割り当てる縮約基底関数ブロック I_B，J_B，K_B の組み合わせ（I_B，J_B，K_B）を決定する（ステップＳ２）。

【００７２】次に、ホスト計算機は、プロセッサエレメントへ、前述の（数式４２）の関係を満たす３個の縮約基底関数ブロック I_B，J_B，K_B に対応して、要素数が各々 $I_C \times I_C$ 個の密度行列要素ブロックＰ（I_B，J_B），Ｐ（I_B，K_B），Ｐ（J_B，K_B）および要素数が各々 $K_B \times I_C \times I_C$ 個の密度行列要素ブロックの列Ｐ（I_B，Ｌ），Ｐ（J_B，Ｌ），Ｐ（K_B，Ｌ）を送信する（ステップＳ３およびステップＳ４）。但し、Ｌは、１から $K_B \times I_C$ の範囲の全てである。

【００７３】次に、これらをステップＳ１１およびステップＳ１２で受信したプロセッサエレメントは、内部でＬに関するループをまわし、単純ブロック法と同じ２電子積分およびフォック行列要素を計算する（ステップＳ１３）。全てのＬに対する計算が終了すると、プロセッサエレメントは、要素数が各々 $K_B \times I_C \times I_C$ 個のフォック行列要素ブロックの列Ｆ（I_B，Ｌ），Ｆ（J_B，Ｌ），Ｆ（K_B，Ｌ）および要素数が各々 $I_C \times I_C$ 個のフォック行列要素ブロックＦ（I_B，J_B），Ｆ（I_B，K_B），Ｆ（J_B，K_B）をホスト計算機に送り返す（ステップＳ１４およびステップＳ１５）。

【００７４】ホスト計算機は、ステップＳ５およびステップＳ６で、それらを受信する。そして、ステップＳ２に戻り、以上の処理を繰り返す。

【００７５】このようにすることにより、密度行列要素やフォック行列要素の利用効率がさらに高まり、２電子積分１つあたりの通信データ数は、$2/N I_C + 2/I_C^2$ ［ｐｅｒＥＲＩ］となり、$N >> I_C$ を仮定すれば、単純ブロック法の約半分となる。

【００７６】さらに、プロセッサエレメント上の I_B，J_B，K_B の組み合わせを更新した際に、変化するのが K_B のみである場合には、行列要素ブロックＰ（I_B，J_B），Ｆ（I_B，J_B）および行列要素ブロックの列Ｐ（I_B，Ｌ），Ｐ（J_B，Ｌ），Ｆ（I_B，Ｌ），Ｆ（J_B，Ｌ）は、プロセッサエレメント上に残して再利用できるので、通信データ数はさらに減って、$4/3N I_C + 2/3 I_C^2$ ［ｐｅｒＥＲＩ］となる。

【００７７】列ブロック法と同様の思想で、さらに密度行列要素やフォック行列要素の利用効率を高める第５の手法として、クラスタリング法が、文献２では紹介されている。しかしながら、この手法は、負荷分散あるいはスケーラビリティの観点から劣る手法であるとされており、ここでは説明を省略する。

【００７８】

【発明が解決しようとする課題】［カットオフを考慮した場合の問題点］上述した文献３に記載されているうちで、最も優れた第４のアルゴリズムであっても、カットオフを考慮した場合には不都合が生じる場合がある。そのような例を以下に示す。なお、カットオフにより生き残る割合を α、プロセッサエレメント数をＭ、１つの２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）当たりの計算時間をＴｅｒｉ（μｓ）、行列要素のデータ長を６４ビットとする。

【００７９】カノニカル法の場合には、１つのプロセッサエレメントに割り当てられたジョブ当たりに発生する通信量は、最低（すなわち、組み合わせの更新時に変化するのが K_B だけの場合）でも、

$2(2 I_C^2 + K_B I_C^2) \times 64$　（ｂｉｔ）

となる一方、その間にプロセッサエレメントで計算される２電子積分の個数は、

$\alpha^2 K_B I_C^4$　（個）

なので、全体で必要な通信性能は、図３２に示す（数式４４）のようなものとなる。

【００８０】トリプルソート法の場合には、１つのプロセッサエレメントに割り当てられたジョブ当たりに発生する通信量は、最低（すなわち、組み合わせの更新時に変化するのが K_B だけの場合）でも、

$2(2 I_C^2 + K_B I_C^2) \times 64$　（ｂｉｔ）

となる一方、その間にプロセッサエレメントで計算される２電子積分の個数は、

$3\alpha^2 K_B I_C^4$　（個）

なので、全体で必要な通信性能は、図３２に示す（数式４５）のようなものとなる。

【００８１】並列処理のプロセッサ・エレメント数Ｍを１００、プロセッサエレメントにおける原始基底関数で表わした２電子積分ｇ（ｉ，ｊ，ｋ，ｌ）の計算時間が２μｓ、縮約基底関数の平均縮約数を１．５（従って、２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）１つ当たりの計算時間Ｔｅｒｉを１０μｓ）、カットオフにより生き残る割合

αを０．０５、とした場合に、必要とされるホスト計算機とプロセッサエレメントとの間の通信性能のブロックサイズI_Cに対する依存性を計算すると、カノニカルの場合には、図２１に示すように、トリプルソートの場合には、図２２に示すようになる。

【００８２】カノニカルの場合にも、また、トリプルソートの場合にも、必要な通信性能はK_Bの値に依存して変化するが、K_B＞１００ではその変化量が小さく、実際の通信性能に応じて、ブロックサイズI_Cを、カノニカルの場合なら、例えば５０に、トリプルソートの場合であれば、例えば３０に、それぞれ設定することが可能である。

【００８３】文献３で前提としているような、十分に大きなデータ保持容量を有するワークステーションを、多数台用いるシステムでは、このような方法で計算を行うことが可能であるが、そのようなシステムを構成するには、多大な費用が必要である。

【００８４】一方、文献２にあるような、せいぜい数１０Ｍビット程度の小さな容量のメモリが接続される安価な専用プロセッサ・エレメントを用いた計算システムでは、列単位で保持できる行列要素は１０列以下、すなわち、許されるブロックサイズは２から３程度までである。

【００８５】この場合には、図２１あるいは図２２の結果から、通信の手段としてＩＥＥＥ１３９４バス規格で定められているような安価なシリアル通信を用いた場合には、その性能が２００Ｍｂｐｓであることを考慮すると、通信性能が律速しない効率的な処理を行うことが不可能である。

【００８６】この発明は、以上のような点にかんがみ、安価な通信手段と小容量のメモリを有する多数のプロセッサ・エレメントを用いた並列計算によっても、ホスト計算機とプロセッサ・エレメントとの間の通信性能に律速されることなく、効率的に行列要素計算を行える並列処理方法を提供することを目的とする。

【００８７】

【課題を解決するための手段】上記目的を達成するために、請求項１の発明の並列処理方法は、同じ１からＮ（Ｎは正の整数）の範囲にある４つの整数型の変数Ｒ，Ｓ，Ｔ，Ｕを用いて表わされ、Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）＝Ｇ（Ｒ，Ｓ，Ｕ，Ｔ）＝Ｇ（Ｓ，Ｒ，Ｔ，Ｕ）＝Ｇ（Ｓ，Ｒ，Ｕ，Ｔ）＝Ｇ（Ｔ，Ｕ，Ｒ，Ｓ）＝Ｇ（Ｔ，Ｕ，Ｓ，Ｒ）＝Ｇ（Ｕ，Ｔ，Ｒ，Ｓ）＝Ｇ（Ｕ，Ｔ，Ｓ，Ｒ）なる関係を満たす関数Ｇの関数値Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）と；２つの変数Ｔ，Ｕを用いて表わされ、Ｐ（Ｔ，Ｕ）＝Ｐ（Ｕ，Ｔ）なる関係を満たす行列Ｐの要素Ｐ（Ｔ，Ｕ）と；係数Ａ１との積Ａ１・Ｐ（Ｔ，Ｕ）・Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）についての前記範囲の全てのＴおよびＵに関する総和Ｆ１（Ｒ，Ｓ）と、前記関数値Ｇ（Ｒ，Ｕ，Ｔ，Ｓ）と；前記行列要素Ｐ（Ｔ，Ｕ）と；

係数Ａ２との積Ａ２・Ｐ（Ｔ，Ｕ）・Ｇ（Ｒ，Ｕ，Ｔ，Ｓ）に関する前記範囲の全てのＴおよびＵにおける総和Ｆ２（Ｒ，Ｓ）との和Ｆ（Ｒ，Ｓ）＝Ｆ１（Ｒ，Ｓ）＋Ｆ２（Ｒ，Ｓ）を要素とする行列Ｆの全要素の計算を、ホスト計算機と、１つまたは複数個のプロセッサエレメントとを有する並列処理装置を用いて行う並列処理方法において、前記変数Ｒ，Ｓ，Ｔ，Ｕについて、３重ループを形成し、前記３重ループの最も外側のループは、Ｒ≦ＮおよびＴ≦Ｒなる関係を満たす変数Ｒと変数Ｔとの組み合わせに関するループとし、前記最も外側のループの内側の２番目は前記変数Ｓに関するループ、前記２番目よりも内側の３番目は前記変数Ｕに関するループとするか、あるいは前記２番目は変数Ｕに関するループ、前記３番目は変数Ｓに関するループとし、前記変数Ｓのとりうる値の範囲を１からＲの間とし、前記変数Ｕのとりうる値の範囲を１からＲの間とし、前記３番目のループの内側で、所定の前記関数値Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）の計算およびその計算結果を用いた所定の前記行列要素Ｆの一部の計算を行うものであって、前記２番目および３番目のループをひとまとまりとして１つのジョブ単位を形成し、前記ジョブ単位毎に前記複数のプロセッサエレメントに処理させることを特徴とする。

【００８８】また、請求項２の発明の並列処理方法は、Ｎ個（Ｎは正の整数）の縮約シェルを用いて表現される分子のエネルギーを計算する分子軌道計算を、ホスト計算機と、１つまたは複数個のプロセッサエレメントとを有する並列処理装置を用いて行う並列処理方法において、縮約シェルＲ，Ｓ，Ｔ，Ｕのそれぞれに含まれる原始シェルｒ，ｓ，ｔ，ｕのそれぞれの成分である原始基底関数ｉ，ｊ，ｋ，ｌをインデックスとして用いて表わされる２電子積分関数ｇの関数値ｇ（ｉ，ｊ，ｋ，ｌ）と；前記原始基底関数ｋをひとつの構成要素とする縮約基底関数Ｋおよび前記原始基底関数ｌをひとつの構成要素とする縮約基底関数Ｌとをインデックスとして用いて表わされる密度行列Ｐの要素Ｐ（Ｋ，Ｌ）と；係数Ａ１との積Ａ１・Ｐ（Ｋ，Ｌ）・ｇ（ｉ，ｊ，ｋ，ｌ）の全ての縮約基底関数に関する総和ｆ１（Ｉ，Ｊ）と、前記２電子積分関数の関数値ｇ（ｉ，ｋ，ｊ，ｌ）と；前記密度行列Ｐの要素Ｐ（Ｋ，Ｌ）と；係数Ａ２との積Ａ２・Ｐ（Ｋ，Ｌ）・ｇ（ｉ，ｋ，ｊ，ｌ）の全ての縮約基底関数に関する総和ｆ２（Ｉ，Ｊ）との和ｆ（Ｉ，Ｊ）＝ｆ１（Ｉ，Ｊ）＋ｆ２（Ｉ，Ｊ）の、前記原始基底関数ｉ，ｊをひとつの構成要素とする前記縮約基底関数Ｉ，Ｊに含まれる全ての前記原始基底関数に関する和で表わされるフォック行列Ｆの全ての行列要素Ｆ（Ｉ，Ｊ）の計算を、最も外側のループは、Ｒ≦ＮおよびＴ≦Ｒなる関係を満たす前記縮約シェルＲとＴとの組み合わせに関するループとし、前記最も外側のループの内側の２番目は前記縮約シェルＳに関するループ、前記２番目よりも内側の３番目は前記縮約シェルＵに関するループ

とするか、あるいは前記２番目は前記縮約シェルUに関するループ、前記３番目は前記縮約シェルSに関するループとし、前記縮約シェルSのとりうる値の範囲を１からRの間とし、前記縮約シェルUのとりうる値の範囲を１からRの間とし、前記３番目のループの内側で、所定の２電子積分の計算およびその結果を用いた所定のフォック行列要素の一部の計算を行うものであって、前記２番目および３番目のループをひとまとまりとして１つのジョブ単位を形成し、前記ジョブ単位毎に前記複数のプロセッサ・エレメントに処理を割り当てることを特徴とする。

【００８９】

【作用】上述の構成の請求項１の発明、請求項２の発明を用いて、フォック行列計算アルゴリズムを行うと、カットオフを考慮して２電子積分の計算数を減少させる場合においても、ホスト計算機とプロセッサエレメントとの間の通信性能およびホスト計算機の処理性能に律速されることのない効率的な並列計算によって、フォック行列を高速に計算することができる。

【００９０】

【発明の実施の形態】以下、この発明の実施の形態を、図を参照しながら説明する。以下に説明する実施の形態では、図１に示すような安価な計算機システムを用いる。

【００９１】すなわち、図１は、この実施の形態の並列計算システムの全体の構成を示すもので、ホスト計算機１に対して、バス３を通じて、複数個のプロセッサエレメント２が接続されている。バス３としては、例えばＩＥＥＥ１３９４シリアルバスが用いられる。

【００９２】なお、各プロセッサエレメント２のメモリ容量は、数１０Ｍビット、例えば２０Ｍビットであり、行列要素が６４ビットの浮動小数点数で表わされる密度行列およびフォック行列のサイズが、１００００×１００００であっても、各行列の１０列分は十分格納できる容量のものが用いられる。この程度の容量のメモリを各プロセッサエレメントに持たせることは、現在の技術で十分に可能である。

【００９３】まず、この実施の形態における行列要素計算方法と比較するために、従来例のトリプルソート法における全体の計算アルゴリズムを、プログラムコードの形式で表わしたものを図２に示す。

【００９４】図２で、Ｎｓｈｅｌｌは縮約シェルの総数を表わす整数型の変数、Ｒ，Ｓ，Ｔ，Ｕは縮約シェル番号に用いる整数型の変数、Ｉ，Ｊ，Ｋ，Ｌは縮約基底番号に用いる整数型の変数、Ｇ＿ＩＪＫＬ，Ｇ＿ＩＫＪＬ，Ｇ＿ＩＬＪＫは２電子積分の数値に用いる実数型の変数、Ｆ，Ｐはフォック行列および密度行列に用いる実数型の２次元配列である。

【００９５】また、ここでは、縮約基底が通し番号で番号付けされていること、および同一の縮約シェルを構成する縮約基底の番号が連続していることを前提として、ｂ＿ｂａｓｉｓおよびｅ＿ｂａｓｉｓは、その引数となっている番号に相当する縮約シェルを構成する縮約基底の番号の始まりと終わりをリターン値とする整数型の関数である。さらに、Ｇは、その引数となっている４つの番号Ｉ，Ｊ，Ｋ，Ｌに対応する縮約基底で一意的に決まる２電子積分を、前述の（数式２３）に従って計算する実数型の関数である。

【００９６】さて、この図２に示す従来例のトリプルソート法では、縮約シェルの計算に関する４重ループのうちの最も内側で行われるフォック行列要素への足し込みの計算において、密度行列要素Ｐ［Ｉ］［Ｊ］，Ｐ［Ｉ］［Ｋ］，Ｐ［Ｉ］［Ｌ］，Ｐ［Ｊ］［Ｋ］，Ｐ［Ｊ］［Ｌ］，Ｐ［Ｋ］［Ｌ］が用いられている。

【００９７】したがって、縮約シェルに関する４重ループをどのような順序で形成しても、ホスト計算機１から密度行列要素がプロセッサ・エレメント２に送信される頻度を、１つの２電子積分計算あたりの行列要素数で表わすと、それは１のオーダー（トリプルソート法では正確には１／３）となる。

【００９８】前述した従来例におけるブロック法は、行列要素の再利用性を高めて密度行列要素が送信される頻度を１より小さい定数倍にするものであった。ここで、その定数は、ブロックサイズの２乗に反比例するものである。図１に示すような安価なシステムを前提とした場合には、プロセッサエレメント２のメモリの容量の大きさで上限が規定されて、ブロックサイズを大きくできないため、密度行列要素が送信される頻度が、１よりも大して小さくならなかった。

【００９９】また、計算の結果として得られるフォック行列要素は、Ｆ［Ｉ］［Ｊ］，Ｆ［Ｉ］［Ｋ］，Ｆ［Ｉ］［Ｌ］，Ｆ［Ｊ］［Ｋ］，Ｆ［Ｊ］［Ｌ］，Ｆ［Ｋ］［Ｌ］であり、プロセッサ・エレメント２からホスト計算機１へフォック行列要素が送信される頻度は、密度行列の場合と全く同様であった。さらに、カットオフを考慮した場合には、カットオフにより生き残る割合αの２乗に反比例して通信の頻度が高くなるという問題があった。

【０１００】この発明の実施の形態のアルゴリズムでは、カットオフにより生き残る割合αに対する通信の頻度の依存性を弱くし、通信量を低く保つことが可能である。この実施の形態の計算アルゴリズムをプログラムコードの形式で表わしたものを図３に示す。

【０１０１】すなわち、図３のアルゴリズムでは、最も外側のループはＲ≦Ｎｓｈｅｌｌおよび Ｒ≧Ｔなる関係を満たす縮約シェルＲとＴとの組み合わせ（ＲＴ）に関するループとしている。そして、２番目は縮約シェルＳに関するループ、３番目は縮約シェルＵに関するループとしている。この場合、Ｓのとりうる値の範囲は１からＲの間とし、また、Ｕのとりうる値の範囲も１からＲの

間とする。さらに、３番目のループの内側で、関数値Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）の計算およびその結果を用いた所定の行列要素Ｆの一部の計算を行うようにしている。

【０１０２】なお、２番目はＵに関するループ、３番目はＳに関するループとするようにしてもよい。すなわち、図３では、Ｕに関するループが、Ｓに関するループの内側に形成されているが、このループの順序は逆でも良い。

【０１０３】そして、ＲとＴとが固定された２番目および３番目の２つのループをひとまとまりとして一つのジョブ単位を形成し、このジョブ単位毎に複数のプロセッサエレメント２に処理させるようにする。

【０１０４】このとき、ホスト計算機１は、全ての行列要素Ｐの初期値の計算と、プロセッサエレメントと共に行うＳＣＦ計算と、このＳＣＦ計算を継続するかどうかの決定などを行う。ＳＣＦ計算に当たって、ホスト計算機１は、複数個のプロセッサエレメント２に対するＲとＴとが固定されたジョブ単位の割り当て決定と、プロセッサエレメントに対して送信すべき行列Ｐの一部の行列要素の選択と、選択された行列要素のプロセッサエレメントに対する送信と、プロセッサエレメントから送信されてきた行列Ｆの一部の行列要素の受信と、行列Ｆを用いた行列Ｐの更新とを行う。

【０１０５】一方、プロセッサエレメント２は、ホスト計算機から送信された行列Ｐの一部の行列要素の受信と、縮約シェルＳに関するループの制御および縮約シェルＵに関するループの制御と、関数Ｇ（Ｒ，Ｓ，Ｔ，Ｕ）または関数ｇ（ｉ，ｊ，ｋ，ｌ）の計算と、行列Ｆの一部の行列要素の計算と、行列Ｆの一部の行列要素のホスト計算機に対する送信とを行う。

【０１０６】この実施の形態では、ホスト計算機１とプロセッサ・エレメント２との間で通信される密度行列要素およびフォック行列要素を、Ｐ［Ｉ］［Ｊ］，Ｐ［Ｉ］［Ｌ］，Ｐ［Ｊ］［Ｋ］，Ｐ［Ｋ］［Ｌ］およびＦ［Ｉ］［Ｊ］，Ｆ［Ｉ］［Ｌ］，Ｆ［Ｊ］［Ｋ］，Ｆ［Ｋ］［Ｌ］だけにし、また、プロセッサ・エレメント２で計算する２電子積分を、Ｇ__ＩＪＫＬだけにした。この実施の形態の場合、行列要素の通信頻度は、１／Ｎのオーダーである。

【０１０７】ここで、図３の計算アルゴリズムで行列要素の通信頻度を１／Ｎのオーダーとすることが可能な理由を説明する。

【０１０８】前述したように、図３において、最も外側のループは、縮約シェルＲとＴの組み合わせ（以下ではＲＴペアと記す）の番号ＲＴに関するループである。ＲとＴの組み合わせの総数は、Ｎｓｈｅｌｌ×（Ｎｓｈｅｌｌ＋１）／２である。以下の説明ではＲ≧Ｔを前提とする。したがって、Ｒの取りうる範囲は、１からＮｓｈｅｌｌまで、Ｔの取りうる範囲は、１からＲまでである。

【０１０９】図３では、表記を簡略にするために、ＲＴに関しては、１からＮｓｈｅｌｌ×（Ｎｓｈｅｌｌ＋１）／２まで、ＳおよびＵに関しては、１からＲまで、それぞれ番号の小さい順番にループを回すようにしているが、必ずしもこのような順序でループを回す必要はない。また、ＳおよびＵに関しては、１からＲまでの全ての値を取る必要はない。

【０１１０】縮約シェルＵに関するループの内部では、縮約基底Ｉ，Ｊ，Ｋ，Ｌに関する４重ループとなっており、その内部で２電子積分の計算を行い、また、Ｉ，Ｊ，Ｋ，Ｌ相互の関係に依存した条件に従ったフォック行列の足し込み計算を行なうアルゴリズムとなっている。上記の条件分岐により、（数式１５）に示したフォック行列要素の計算を、過不足なく計算することが可能となる。

【０１１１】前述したように、図３のアルゴリズムで、縮約シェルＲおよびＴに関するループの制御はホスト計算機１で行い、縮約シェルＳに関するループより内側はプロセッサエレメント２で行う。すなわち、各プロセッサエレメント２には、縮約シェルＲとＴの組み合わせ（ＲＴペア）が固定されたジョブが割り当てられる。あるＲＴペアが固定されたジョブが割り当てられたプロセッサ・エレメント２において、使用する密度行列要素は、Ｐ［Ｉ］［Ｊ］，Ｐ［Ｉ］［Ｌ］，Ｐ［Ｊ］［Ｋ］，Ｐ［Ｋ］［Ｌ］である。

【０１１２】ここで、ＩおよびＫは、それぞれ縮約シェルＲおよびＴを形成する縮約基底であるので、ジョブが割り当てられた時点で、これらは数個（縮約シェルがｓ軌道の場合には１個、ｐ軌道の場合には３個、ｄ軌道の場合には６個）に固定される。ＪおよびＬは任意である。

【０１１３】したがって、ＲＴペアに関するジョブのために、ホスト計算機１からプロセッサエレメント２に送信する密度行列要素数は、Ｎの定数倍のオーダーである。フォック行列に関しても同様である。プロセッサエレメント２には、これら全てを保持することができるだけの容量のメモリを有している。

【０１１４】一方、ＲＴペアが固定されたジョブが割り当てられたプロセッサエレメントでは、縮約シェルＳおよびＵに関するループが回るので、そこで計算される２電子積分の個数は、N^2 のオーダーとなる。したがって、１つの２電子積分当たりに換算した通信の頻度は１／Ｎとなる。

【０１１５】このように、複数のプロセッサエレメント２へのジョブ割り当てが、ＲＴペア単位で行われることから、この発明のアルゴリズムを、ＲＴ並列アルゴリズムと呼ぶことにする。

【０１１６】ＲＴ並列アルゴリズムでカットオフを考慮する場合には、プロセッサエレメント２で計算する２電子積分の個数は、カットオフにより生き残る割合αの２

乗に比例して減少する。一方、計算に用いる密度行列要素および計算されるフォック行列要素も、カットオフにより生き残る割合αに比例して減少させることができる。この理由を以下に説明する。

【０１１７】プロセッサエレメント２に割り当てるＲＴペアを決めた時点で、ホスト計算機１は、ＩＪペアでカットオフされないＪ、およびＫＬペアでカットオフされないＬをリストアップできる。

【０１１８】したがって、ホスト計算機１は、Ｉが固定された１列の密度行列要素Ｐ［Ｉ］［Ｊ］の中からＩＪペアでカットオフされないＪに対応するものだけを、Ｋが固定された１列の密度行列要素Ｐ［Ｋ］［Ｊ］の中からＩＪペアでカットオフされないＪに対応するものだけを、Ｉが固定された１列の密度行列要素Ｐ［Ｉ］［Ｌ］の中からＫＬペアでカットオフされないＬに対応するものだけを、Ｋが固定された１列の密度行列要素Ｐ［Ｋ］［Ｌ］の中からＫＬペアでカットオフされないＬに対応するものだけを、それぞれ選び出して送信することが可能である。そのため、ホスト計算機１から送信される密度行列要素数は、αに比例して減少する。

【０１１９】また、プロセッサエレメント２は、Ｉが固定された１列のフォック行列要素Ｆ［Ｉ］［Ｊ］のうち、ＩＪペアでカットオフされないＪに対応するものだけを、Ｋが固定された１列のフォック行列要素Ｆ［Ｋ］［Ｊ］のうち、ＩＪペアでカットオフされないＪに対応するものだけを、Ｉが固定された１列のフォック行列要素Ｆ［Ｉ］［Ｌ］のうち、ＫＬペアでカットオフされないＬに対応するものだけを、Ｋが固定された１列のフォック行列要素Ｆ［Ｋ］［Ｌ］のうちＫＬペアでカットオフされないＬに対応するものだけを、それぞれ計算する。

【０１２０】プロセッサエレメント２からホスト計算機１へは、計算されたフォック行列要素のみを送信すればよいので、プロセッサエレメント２からホスト計算機１へ送信されるフォック行列要素数も、カットオフにより生き残る割合αに比例して減少する。

【０１２１】したがって、カットオフを考慮した計算を行う場合においては、１つの２電子積分当たりに換算した通信の頻度は、カットオフにより生き残る割合αに反比例して増加するが、それは、αの２乗に反比例して増加する従来例と比較して、実用上の大きな利点となる。

【０１２２】次に、プロセッサエレメントの数をＭ、１つの２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）当たりの計算時間をＴｅｒｉ（μｓ）、１つの縮約シェルを構成する平均の縮約基底数をａ、行列要素のデータ長を６４ｂｉｔとして、必要な通信性能を定式化する。

【０１２３】密度行列要素ブロックＰ（Ｒ，Ｓ）が、縮約シェルＲを構成する任意の縮約基底Ｉと縮約シェルＳを構成する任意の縮約基底Ｊとの組み合わせで表わされる密度行列要素Ｐ（Ｉ，Ｊ）の集合であるとすると、その密度行列要素ブロックの要素数は、約a^2となる。密度行列要素ブロックＰ（Ｒ，Ｓ）は、Ｓ＝１から、Ｓ＝Ｒまでの範囲のうち、ＲＳの組み合わせでのカットオフで生き残るもの全てが、ホスト計算機１からプロセッサエレメント２へ転送される。したがって、その要素数は、$a^2 \times aR \times \alpha$個となる。

【０１２４】他の密度行列要素ブロックＰ（Ｒ，Ｕ），Ｐ（Ｔ，Ｓ），Ｐ（Ｔ，Ｕ）に関しても、概略で同じ要素数となる。また、プロセッサエレメント２からホスト計算機１へ送信するフォック行列要素数は、密度行列要素数と全く同じである。したがって、１つのジョブ当たりのホスト計算機１とプロセッサエレメント２との間の通信データ量は、

$8\alpha a^3 R \times 64$　（ビット）

となる。

【０１２５】一方、１つのジョブ当たりで計算される２電子積分の個数は、概略で、

$\alpha^2 a^4 R^2$　（個）

なので、全体で必要な通信性能は、図３２の（数式４６）に示すようなものと見積もられる。

【０１２６】プロセッサエレメント２の数Ｍを１００、プロセッサエレメント２における原始基底関数で表わした２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）当たりの計算時間Ｔｅｒｉを１０（μｓ）、１つの縮約シェルを構成する平均の縮約基底数ａを３、カットオフによる生き残り割合αを０．０５とした場合に、必要な通信性能のＲに対する依存性を図４に示す。

【０１２７】この図４からわかるように、縮約シェルＲが小さいごく一部の範囲では、必要な通信性能が１００Ｍｂｐｓを超えるが、他の殆どの領域では１００Ｍｂｐｓの通信性能があれば、十分に通信時間を計算時間でカバーできることがわかる。

【０１２８】一定のＲに対するジョブ数が、R^2に比例することを考慮すると、通信時間が計算時間を上回る可能性は、ごくわずかとなる。したがって、この実施の形態のＲＴ並列アルゴリズムは、１００Ｍｂｐｓ程度の通信性能の、安価な通信手段を用いても、システム全体の処理効率を低下させることのない、効率的なアルゴリズムとなっている。

【０１２９】［より具体的な説明］以下では、この発明の実施の形態の並列処理方法の詳細を、より具体的に説明する。

【０１３０】［ホスト計算機１の処理手順］図５は、この発明の実施の形態の並列処理方法を説明するフローチャートである。ホスト計算機１とプロセッサエレメント２とでは異なる処理を行うため、それぞれについてフローチャートを併記した。

【０１３１】また、プロセッサエレメント２は、並列に複数（典型的には１００個）がホスト計算機１に接続されていることを前提としているが、それらは同様のフロ

ーチャートに従って動作し、また、それらの間のデータの相関はないので、代表として１つのプロセッサエレメント２における処理フローを表記した。なお、図５において、破線で示した矢印の部分は、その後の処理が、その前の処理で律速されるのでなく、他の処理系からの情報の入力待ちとなることを示している。

【０１３２】まず、ホスト計算機１の処理手順を説明する。ホスト計算機１は、係数行列の初期設定（ステップＳ１０１）を行った後、（数式１８）に従って係数行列から密度行列を計算する（ステップＳ１０２）。次に、特定のプロセッサエレメント２に割り当てる縮約シェル番号ＲとＴの組み合わせ、すなわち（ＲＴ）ペア番号を決める（ステップＳ１０３）。（ＲＴ）ペア番号の割り当て順序は任意であり、ランダムに決めても良いし、ある一定の規則に従って決めても良い。

【０１３３】次に、（ＲＴ）ペア番号に対応する密度行列情報を、その（ＲＴ）ペア番号が割り当てられたプロセッサエレメントに対して送信する（ステップＳ１０４）。送信される密度行列情報は、カットオフ条件を考慮して作成されるが、そのときにホスト計算機１が行う処理内容に関しては後述する。

【０１３４】プロセッサエレメント２は、このホスト計算機からの密度行列情報を受信し、受信バッファメモリに格納する（ステップＳ２０１）。そして、割り当てられた（ＲＴペア）のジョブについて、縮約シェルＳ，Ｕに関するループを制御して、フォック行列要素の計算を行う（ステップＳ２０２）。そして、計算が終了すると、求めたフォック行列情報を送信バッファメモリに格納し、その送信バッファメモリからホスト計算機１に、その求めたフォック行列情報を送信する（ステップＳ２０３）。

【０１３５】ホスト計算機１は、以上のようにして、あるプロセッサエレメント２での処理が終了して、そのプロセッサエレメント２からフォック行列情報を受信すると（ステップＳ１０５）、そのプロセッサエレメント２に対する全てのプロセッサ・エレメントに対する（ＲＴ）ペア番号の割り当てと密度行列情報の送信が終了しかたどうか判断し（ステップＳ１０７）、終了していなければ、ステップＳ１０３に戻って、新たな（ＲＴ）ペア番号の割り当てを決め、その（ＲＴ）ペア番号に対応する密度行列情報を、そのプロセッサエレメント２に送信する。

【０１３６】同様の操作を繰り返して、ホスト計算機１は、全てのプロセッサエレメントに対する（ＲＴ）ペア番号の割り当てと密度行列情報の送信を行い、プロセッサエレメントからのフォック行列の受信を待つ。

【０１３７】ステップＳ１０６で、全ての（ＲＴ）ペア番号の処理が終了したと判断すると、ホスト計算機１は、収集したフォック行列情報をもとに、図２６の（数式１４）を解き、新たな係数行列を計算する（ステップＳ１０７）。そして、新たな係数行列とその直前の係数行列とを比較して、自己無撞着な解が得られたかどうか判断する（ステップＳ１０８）。そして、両者が十分良く一致していれば、自己無撞着な解が得られたと判断し、計算を終了する。そうでなければ、ステップＳ１０８からステップＳ１０２に戻り、（数式１８）に従って新たな密度行列を生成し、同様の操作を繰り返す。

【０１３８】ここで、ホスト計算機１におけるカットオフ処理について説明を加える。

【０１３９】縮約シェルＲが与えられ、その原子核座標を、その縮約シェルＲを構成する原始基底の軌道指数の最小値をζ_A、その原始基底をｉとする。また、任意の縮約シェルＳの原子核座標を、その縮約シェルＳを構成する原始基底の軌道指数の最小値をζ_B、その原始基底をｊとする。

【０１４０】従来の技術の欄で記したように、任意の原始基底ｋ，ｌに対して、２電子積分ｇ（ｉ，ｊ，ｋ，ｌ）をカットオフできるかどうかが、（ζ_A，Ａ，ζ_B，Ｂ）の数値の組み合わせによって判断できる。例えば、図３２の（数式４７）を満足するかどうかを判定し、この（数式４７）の不等式が成り立っていれば、カットオフできると判断する。

【０１４１】それぞれ同じ縮約シェルに属する任意の２つの原始基底の組み合わせで考えると、これらの軌道指数は、上記のものよりも大きいため、その組み合わせで計算した指数関数の数値は、上記のものよりも必ず小さくなる。

【０１４２】したがって、この最小の軌道指数の組み合わせでカットオフできると判断された場合、縮約シェルＲを構成する任意の原始基底と、縮約シェルＳを構成する任意の原始基底との組み合わせで、必ずカットオフできることになる。

【０１４３】このように、縮約シェルＳとして全ての縮約シェルを当てはめて判定を行い、縮約シェルＲとのペアでのカットオフに生き残る縮約シェルの番号を１つの集合とすることができる。

【０１４４】ホスト計算機１は、ＳＣＦ計算の開始前に、全ての縮約シェルに対して、それとのペアでのカットオフに生き残る番号の集合を、データベースとして作成しておく。これを、この実施の形態では、カットオフテーブルと呼ぶことにする。

【０１４５】図６に、このカットオフテーブルの一例を示す。この例において、例えば、番号が１の縮約シェルが、Ｒとして選ばれた（Ｒ＝１）場合、カットオフで生き残る縮約シェルＳの番号は１，２，３，５となる。但し、ＲＴ並列アルゴリズムで、プロセッサエレメント２において計算が行われるのは、Ｓ≦Ｒの場合のみであるので、Ｒ＝１が割り当てられたプロセッサエレメント２で計算が行われるＳの番号は、１のみとなる。

【０１４６】別の例として、Ｒ＝４の場合を考えると、

カットオフで生き残り、しかも、Ｓ≦Ｒを満たすＳの番号は、２，３，４となる。

【０１４７】図６のカットオフテーブルは、ＴＵの組み合わせにおけるカットオフ判断にも全く同様に用いることができる。例えば、Ｔ＝１の縮約シェルに対して、カットオフで生き残る縮約シェルＵの番号は、１，２，３，５である。ただし、ＲＴ並列アルゴリズムで、プロセッサエレメント２において計算が行われるのは、Ｕ≦Ｒの場合であるので、Ｔ＝１とともに、プロセッサエレメント２に割り当てられたＲの番号に依存して、プロセッサエレメント２で計算が行われるＵの番号は変わる。

【０１４８】例えば、Ｒ＝１０であれば、Ｕとして、１，２，３，５が用いられるし、Ｒ＝２ならば、Ｕとして、１，２が用いられる。

【０１４９】さて、ＳＣＦ計算が開始され、あるプロセッサエレメント２に割り当てるＲＴペア番号が決まると、ホスト計算機１は、このカットオフテーブルを参照しながら密度行列情報を作成する。

【０１５０】縮約シェルＲ，Ｓ，Ｔ，Ｕを構成する縮約基底を、Ｉ，Ｊ，Ｋ，Ｌで、それぞれ表わすことにすると、ホスト計算機１からプロセッサエレメント２へ送信する密度行列は、Ｐ（Ｉ，Ｊ），Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ），Ｐ（Ｋ，Ｊ）であり、また、プロセッサエレメント２で計算される２電子積分は、Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）である。

【０１５１】Ｐ（Ｉ，Ｊ）が必要かどうかは、Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）が、カットオフで生き残るかどうかに依存し、従って、カットオフテーブルに記述されている番号から、Ｓとして用いられる番号をリストアップし、それらの縮約シェルを構成する縮約基底をＪとしてＰ（Ｉ，Ｊ）を選び出せば良い。

【０１５２】このように、Ｊに対して、Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）がカットオフで生き残るかどうかは、ＲＳの組み合わせでカットオフされないかどうかで決まるので、Ｐ（Ｋ，Ｊ）が必要かどうかも全く同様に決めることができる。すなわち、Ｐ（Ｉ，Ｊ）とＰ（Ｋ，Ｊ）とでは、送信すべきＪの番号の集合は全く同じである。

【０１５３】したがって、Ｓ≦Ｒを満たすＲ個の縮約シェルの中で、ＲＳの組み合わせで生き残るＮｖ個の縮約シェルの集合を、Ｖ＝｛Ｖ［１］，Ｖ［２］，・・・，Ｖ［Ｎｖ］｝とすると、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）に係わるものとして、Ｖの要素の個数Ｎｖと、Ｖの要素すなわちＶ［１］，Ｖ［２］，・・・，Ｖ［Ｎｖ］と、Ｖ［ｘ］（ｘ＝１～Ｎｖ）に係わる密度行列データブロックとを、ホスト計算機１からプロセッサエレメント２へ送信する密度行列情報に含めれば良い。

【０１５４】ここで、Ｖ［ｘ］に係わる密度行列データブロックの内容は、縮約シェルＲを構成する全ての縮約基底Ｉと縮約シェルＶ［ｘ］とを構成する全ての縮約基底Ｍの任意の組み合わせに対するＰ（Ｉ，Ｍ）の数値、および縮約シェルＴを構成する全ての縮約基底Ｋと縮約シェルＶ［ｘ］とを構成する全ての縮約基底Ｍの任意の組み合わせに対するＰ（Ｋ，Ｍ）の数値であれば良い。

【０１５５】全く同様に、用いられる縮約シェルＵの番号を、カットオフテーブルからリストアップし、プロセッサエレメント２で用いられるＰ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）を選び出すことが可能である。

【０１５６】Ｕ≦Ｒを満たすＲ個の縮約シェルの中でＴＵの組み合わせで生き残るＮｗ個の縮約シェルの集合を、Ｗ＝｛Ｗ［１］，Ｖ［２］，・・・，Ｗ［Ｎｗ］｝とすると、Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）に係わるものとして、Ｗの要素の個数Ｎｗと、Ｗの要素すなわちＷ［１］，Ｗ［２］，・・・，Ｗ［Ｎｗ］と、Ｗ［ｘ］（ｘ＝１～Ｎｗ）に係わる密度行列データブロックとを、ホスト計算機１からプロセッサエレメント２へ送信する密度行列情報に含めれば良い。

【０１５７】ここで、Ｗ［ｘ］に係わる密度行列データブロックの内容は、縮約シェルＲを構成する全ての縮約基底Ｉと縮約シェルＷ［ｘ］とを構成する全ての縮約基底Ｎの任意の組み合わせに対するＰ（Ｉ，Ｎ）の数値、および縮約シェルＴを構成する全ての縮約基底Ｋと縮約シェルＷ［ｘ］とを構成する全ての縮約基底Ｎの任意の組み合わせに対するＰ（Ｋ，Ｎ）の数値であれば良い。

【０１５８】［転送データの形式］次に、密度行列情報およびフォック行列情報の内容に関して説明する。

【０１５９】図７に、ホスト計算機１からプロセッサエレメント２へ送信する密度行列情報のフォーマットの一例を示す。縮約シェルＲの番号から縮約シェルＷ［Ｎｗ］の番号までは、整数型データである。また、縮約シェルＶ［１］に関わる密度行列データブロックから縮約シェルＷ［Ｎｗ］に関わる密度行列データブロックは、１つまたは複数の浮動小数点型データ、または、固定小数点型データにより構成されるブロックである。

【０１６０】図７の上から２つの縮約シェルＲおよび縮約シェルＴの番号により、その密度行列情報を受け取ったプロセッサエレメント２が処理すべきＲＴペア番号が決まる。ここで、Ｔ≦Ｒであるので、図３２の（数式４８）に示す式によって、ＲＴペア番号（ＲＴ）を一意的に決め、２つの縮約シェル番号を用いる代わりに、１つの整数型データ（ＲＴ）によって、ＲＴペア番号を決めることも可能である。

【０１６１】このようにすることで、転送すべき情報量を、整数データ１つ分だけ減らすことも可能である。しかしながら、密度行列情報全体のデータ量と比較して、整数データ１つの差は、無視できる程度に小さい上、上記のようにして決めた整数データ（ＲＴ）から縮約シェルＲおよびＴの番号を復元するために、プロセッサエレメント２上で行うべき処理が増加するため、図７に示すように、縮約シェルＲおよびＴの番号を、別々に記述する形式を用いるのが、より望ましい。

【０１６２】図７において、縮約シェルＲおよびＴの番号の次は、縮約シェルの集合Ｖおよび縮約シェルの集合Ｗの要素数を表わす２つの整数型データである。

【０１６３】密度行列データブロックの構成例を、縮約シェルＶ［１］に関わる密度行列データブロックを代表例として図８に示す。

【０１６４】縮約シェルの軌道量子数が１より大きい場合には、その縮約シェルに関わる密度行列データブロックは、さらに縮約基底に関わるサブブロックにより構成されることになる。図８に示した例は、縮約シェルＶ［１］を構成する縮約基底が、Ｍ１（Ｖ［１］），Ｍ２（Ｖ［１］），・・・，Ｍｍ（Ｖ［１］）と、ｍ個ある場合であり、それぞれの縮約基底に対応したサブブロックがｍ個ある。

【０１６５】１つのサブブロックは、さらに２つの領域に別れる。１つは、縮約シェルＲを構成する縮約基底Ｉ１（Ｒ）からＩｉ（Ｒ）と、Ｍ１（Ｖ［１］）とをインデックスとする密度行列要素の領域である。もう１つは、縮約シェルＴを構成する縮約基底Ｋ１（Ｔ）からＫｋ（Ｔ）と、Ｍ１（Ｖ［１］）とをインデックスとする密度行列要素の領域である。ここで、ｉ，ｋは、それぞれ縮約シェルＲ，Ｔを構成する縮約基底の個数である。これらの領域に含まれる密度行列要素の個数は、縮約シェルＲおよびＴの軌道量子数によって決まり、したがって、１組の密度行列情報の中では、どのサブブロックも同じ大きさとなる。

【０１６６】図９に、プロセッサエレメント２からホスト計算機１に送信するフォック行列情報のフォーマットの一例を示す。上から２つの縮約シェルＲの番号および縮約シェルＴの番号のみが整数型データであり、それ以降の、縮約シェルＶ［１］に関わるフォック行列データブロックから縮約シェルＷ［Ｎｗ］に関わるフォック行列データブロックは、１つまたは複数の浮動小数点型データまたは固定小数点型データにより構成されるブロックである。

【０１６７】密度行列情報の場合と同様に、上から２つの縮約シェルＲおよび縮約シェルＴの番号により、そのフォック行列情報を送信したプロセッサエレメント２が処理したＲＴペア番号を、ホスト計算機１が認識することができる。また、これらは、ＲとＴから一意に決まる１つの整数型データであるＲＴペア番号で代用することも可能である。

【０１６８】さらに、また、ホスト計算機１上に、プロセッサエレメント２に割り当てたＲＴペア番号を記録しておき、ＲＴペアを表わす番号の代わりに、プロセッサエレメント認識番号により、プロセッサエレメント２が送信したフォック行列情報が、どのＲＴペアに対応するものかを、ホスト計算機１に判断させることも可能である。

【０１６９】フォック行列データブロックの構成は、図８に示した密度行列データブロックと同様である。なお、フォック行列情報には、密度行列情報を構成している、縮約シェルの集合ＶおよびＷの要素数や縮約シェルの番号情報を含む必要がない。なぜならば、ホスト計算機１上にはカットオフテーブルがあるため、ホスト計算機１は、ＲおよびＴから、これらの情報を簡単に再生することが可能だからである。

【０１７０】ホスト計算機１上に十分なデータ保持容量がある場合には、プロセッサエレメント２へ密度行列情報を送信してから、プロセッサエレメント２からフォック行列情報を受信するまでの間、これらの情報を保持しておくことができる。いずれにしても、プロセッサエレメント２からホスト計算機１へ送信するフォック行列情報に、縮約シェルの集合ＶおよびＷの要素数や縮約シェルの番号情報を含む必要はなく、これらをフォック行列情報に含まないことによって、プロセッサエレメント２とホスト計算機１との間の通信データ量を小さくすることができる。

【０１７１】図１０に、密度行列情報およびフォック行列情報のプロセッサエレメント２内のメモリ空間への割付の一例を示す。

【０１７２】図１０では、プロセッサエレメント２のメモリ空間へのアドレッシングが、全てワード・アドレッシングで行われることを前提としている。また、縮約シェルの集合Ｖの要素数を１０００、Ｗの要素数を５００とした場合を示している。

【０１７３】０番地および１番地に、そのプロセッサエレメント２に割り当てられたＲおよびＴの番号を格納する。２番地および３番地には、縮約シェルの集合ＶおよびＷの要素数Ｎｖ，Ｎｗを格納する。４番地から（４＋Ｎｖ－１）番地は、縮約シェルＶ［１］からＶ［Ｎｖ］の番号を格納する領域、（４＋Ｎｖ）番地から（４＋Ｎｖ＋Ｎｗ－１）番地は、縮約シェルＷ［１］からＷ［Ｎｗ］の番号を格納する領域である。以上には全て整数型のデータが保存される。

【０１７４】空き領域を挟んで、２０００番地から９９９９番地は、縮約シェルＶ［１］からＶ［Ｎｖ］に係わる密度行列データブロックを格納する領域、１００００番地から１７９.９９番地は、縮約シェルＷ［１］からＷ［Ｎｗ］に係わる密度行列データブロックを格納する領域である。これらの領域に、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）を、それぞれ４０００個ずつ格納することが可能である。

【０１７５】１８０００番地から２５９９９番地は、縮約シェルＶ［１］からＶ［Ｎｖ］に係わるフォック行列データブロックを格納する領域、２６０００番地から３３９９９番地は、縮約シェルＷ［１］からＷ［Ｎｗ］に係わるフォック行列データブロックを格納する領域である。これらの領域に、Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ），Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）を、それぞれ４０００個ずつ

格納することが可能である。

【０１７６】メモリ上で、１ワードが６４ビットであると仮定しても、これらのデータ領域の容量は、せいぜい２Ｍビット程度である。カットオフで生き残るデータのみが格納されることを考慮すると、数１０００基底以上の規模の分子軌道計算に対しても、十分に必要なデータを格納する領域がある。なお、この例よりも大きなデータ保持能力を有するメモリを実装することも、現在の技術では非常に安価に実現できる。

【０１７７】［ループの回しかた（プロセッサエレメント２の処理手順）］次に、プロセッサエレメント２における処理の手順を、図１１および図１２のフローチャートに従って説明する。

【０１７８】まず、プロセッサエレメント２は、ホスト計算機１から密度行列情報を受信し、それをメモリに格納する（ステップＳ３０１）。格納が済んだ時点から、図５に示したＳおよびＵに関するループによる計算を開始する。

【０１７９】Ｓに関するループは、以下のように制御される。まず、整数型変数ｖを用意し、初期値をゼロにセットする（ステップＳ３０２）。次に、ループに入り、変数ｖを１だけ増加させる（ステップＳ３０３）。次に、メモリから縮約シェルＶ［ｖ］の数値を読み出し、それをＳに代入する（ステップＳ３０４）。

【０１８０】以下、適当な処理を終了すると、変数ｖと、縮約シェルＶの個数Ｎｖとを比較する（図１２のステップＳ３２３）。変数ｖとＮｖとが等しければ、Ｓに関するループを終了し、フォック行列情報をホスト計算機１へ送信して（ステップＳ３２４）、割り当てられたＲＴペアの処理を終了する。その後、プロセッサエレメント２は密度行列情報の受信待ちの状態となる。変数ｖとＮｖとが等しくなければ、ステップＳ３０３で変数ｖを１だけ増加させて、同様の処理を繰り返す。

【０１８１】同様に、Ｕに関するループは以下のように制御される。Ｓに関するループ内で、整数型変数ｗを用意し、初期値をゼロにセットする（ステップＳ３０５）。次に、ループに入り、変数ｗを１だけ増加させる（ステップＳ３０６）。次に、メモリから縮約シェルＷ［ｗ］の数値を読み出し、それをＵに代入する（ステップＳ３０７）。

【０１８２】以下、適当な処理を終了すると、変数ｗと縮約シェルＷの個数Ｎｗとを比較する（ステップＳ３２２）。変数ｗとＮｗとが等しければ、Ｕに関するループを終了し、前記ステップＳ３２３の変数ｖとＮｖとが等しいかどうかの判断を行う。ステップＳ３２２で、変数ｗとＮｗとが等しくなければ、ステップＳ３０６に戻って、変数ｗを１だけ増加させて、同様の処理を繰り返す。

【０１８３】このように、ＳおよびＵに関するループは、メモリに格納された縮約シェルＶ［１］，Ｖ［２］，・・・，Ｖ［Ｎｖ］や、縮約シェルＷ［１］，Ｗ［２］，・・・，Ｗ［Ｎｗ］の数値を、順次、読み出して、それらをＳおよびＵの数値として用いることにより制御され、その結果、プロセッサエレメント２は、カットオフで生き残るＳおよびＵのみに関する数値にしかセットされないため、非常に効率よくループ制御を行うことができる。

【０１８４】分子軌道計算における２電子積分およびフォック行列要素の計算は、縮約シェル単位でなく、縮約基底を単位として行う。そのため、Ｕに関するループの内側では、縮約シェルＲを構成する縮約基底Ｉに関するｂ＿ｂａｓｉｓ（Ｒ）からｅ＿ｂａｓｉｓ（Ｒ）までのループ（ステップＳ３０８、ステップＳ３０９およびステップＳ３２１参照）、縮約シェルＳを構成する縮約基底Ｊに関するｂ＿ｂａｓｉｓ（Ｓ）からｅ＿ｂａｓｉｓ（Ｓ）までのループ（ステップＳ３１０、ステップＳ３１１およびステップＳ３２０参照）、縮約シェルＴを構成する縮約基底Ｋに関するｂ＿ｂａｓｉｓ（Ｔ）からｅ＿ｂａｓｉｓ（Ｔ）までのループ（ステップＳ３１２、ステップＳ３１３およびステップＳ３１９参照）、縮約シェルＵを構成する縮約基底Ｌに関するｂ＿ｂａｓｉｓ（Ｕ）からｅ＿ｂａｓｉｓ（Ｕ）までのループ（ステップＳ３１４、ステップＳ３１５およびステップＳ３１８参照）よりなる４重ループを形成し、その中で、２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）の計算（ステップＳ３１６）、および２電子積分と密度行列要素との積をフォック行列要素に足し込む計算（ステップＳ３１７）を行う。

【０１８５】ここで、同一の縮約シェルを構成している縮約基底の番号が、連続となるように縮約基底の番号付けがなされていることを前提として、縮約シェルＸを構成する縮約基底の番号が、ｂ＿ｂａｓｉｓ（Ｘ）からｅ＿ｂａｓｉｓ（Ｘ）の範囲にあるものとした。

【０１８６】この４重ループの中では、カットオフとして上記のＲＳおよびＴＵの組み合わせによるもの以外を考えない限りは、必ず、２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）の計算が行われる。

【０１８７】したがって、ＳおよびＵのループがまわる間に、メモリに格納された密度行列要素の全てが、必ず、フォック行列要素への足し込み計算に用いられる。計算で用いられないような密度行列要素の通信を行うことは、通信量を無意味に増加させることになるが、そのような無駄な通信が生じないようになっていることにより、低価格な通信手段を用いた場合においてさえ、システムの処理性能が通信性能により制限されることがなく、システムの処理効率を向上させることが可能である。

【０１８８】［データ転送のタイミングおよび行列がメモリに入りきらない場合の方法］縮約シェルＳおよびＵに関するループのうち、外側にあるＳに係わる密度行列

要素Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）やフォック行列要素Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）は、２電子積分Ｇ（Ｉ，Ｊ，Ｋ，Ｌ）の計算終了後に、その都度必要な密度行列要素を、ホスト計算機から受信し、計算し終わったフォック行列要素を、その都度ホスト計算機へ送信する、という手順で送受信することも可能である。

【０１８９】しかしながら、データの通信時には、純粋に通信するデータだけでなく通信プロトコルに関係する付加的な情報が必ず付随するため、通信する情報の単位を細かくすることにより、通信量を増大させることに繋がる。

【０１９０】この実施の形態のように、ＲＴペアにつき、密度行列情報およびフォック行列情報を、それぞれ１回の通信で、ホスト計算機とプロセッサ・エレメントとの間で通信することにより、プロトコルに付随する付加的な情報による通信量の増大を最小限にとどめることができる。

【０１９１】なお、計算の対象とする系のサイズが大きかったり、カットオフ判断を行うための閾値が大きく、カットオフによる生き残り割合が高い場合には、密度行列要素Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）、フォック行列要素Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）、および密度行列要素Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）、フォック行列要素Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）の全てをプロセッサエレメント２内のメモリに保存できなくなることがある。その場合の処理方法を説明する。

【０１９２】まず、密度行列要素Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）と、フォック行列要素Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）とが、全てプロセッサエレメント２内のメモリに保存可能である場合には、これら全てと、密度行列要素Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）およびフォック行列要素Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）がメモリに保存できるように、縮約シェルＵの番号を区切る。

【０１９３】このことにより、（ＲＴ）ペアで指定されるジョブは、複数のサブジョブに分割される。これらのサブジョブは、同一のプロセッサエレメント２に割り当てられるようにした上で、密度行列要素Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）は、最初のサブジョブの開始前に、１度だけホスト計算機１からプロセッサエレメント２に送信する。

【０１９４】これらの密度行列要素は、縮約シェルＵの値が、いくらであろうとも、全て共通に用いることができるからである。一方、密度行列要素Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）は、各サブジョブの開始前に、ホスト計算機１からプロセッサエレメント２に送信し、その前のサブジョブで用いられたメモリ上の密度行列要素Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）の領域に上書きしてしまって構わない。これらの密度行列要素は、特定のＵに係わる計算にしか用いられないからである。

【０１９５】同様に、フォック行列要素Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）を、各サブジョブの終了後に、プロセッサエレメント２からホスト計算機１に送信し、そのメモリ領域を、次のサブジョブで計算されるフォック行列要素Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）の格納領域として開放する。

【０１９６】一方、フォック行列要素Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）は、全てのサブジョブで共通に用いることができるので、全てのサブジョブの終了後に、１度だけプロセッサエレメント２からホスト計算機１へ送信する。

【０１９７】密度行列要素Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）とフォック行列要素Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）との全ては、プロセッサエレメント２内のメモリに保存することができないが、密度行列要素Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）と、フォック行列要素Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）との全てを、プロセッサエレメント２内のメモリに保存可能な場合には、縮約シェルＳの番号を区切って、ジョブを複数のサブジョブに分割する。

【０１９８】このような処理方法とすることにより、全ての行列要素を、プロセッサエレメント２内のメモリに保存可能な場合と比較した通信量の増加分は、密度行列要素とフォック行列要素とを分割して送受信することに起因する通信プロトコルに関係する付加的な情報の個数の増分のみとなる。

【０１９９】密度行列要素Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）とフォック行列要素Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）との全て、および密度行列要素Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）とフォック行列要素Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）との全てのどちらも、プロセッサエレメント２内のメモリに格納できないような、計算規模が非常に大きい場合に対しても、この実施の形態のＲＴ並列アルゴリズムは対応可能である。以下に、その処理アルゴリズムを説明する。

【０２００】密度行列要素Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）とフォック行列要素Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）（以下では行列情報Ａとする）との全てで、データサイズがＸであり、密度行列要素Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）とフォック行列要素Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）（以下では行列情報Ｂとする）との全てで、データサイズがＹであると仮定する。

【０２０１】また、縮約シェル番号Ｓを区切って、行列情報Ａをｘ等分し、縮約シェル番号Ｕを区切って、行列情報Ｂをｙ等分したときに、Ｘ／ｘ＋Ｙ／ｙのサイズのデータ量であれば、プロセッサエレメント２内のメモリに保存可能であると仮定する。（ＲＴ）ペアで指定される１つのジョブが、ｘ×ｙ個のサブジョブに分割されたことになる。

【０２０２】図１３に示すように、サブジョブの番号を（Ｓに関する分割番号、Ｕに関する分割番号）の形式で表わし、（１，１）から（１，ｙ）をプロセッサエレメントＰＥ１に、（２，１）から（２，ｙ）をプロセッサエレメントＰＥ２に、（ｘ，１）から（ｘ，ｙ）をプロ

セッサエレメントＰＥｘに、というようにプロセッサエレメントへの割り付けを行う（第１の分割方法）ことにすると、１つのプロセッサエレメントに係わって、ホスト計算機１との間で送受信されるデータ量は、Ｘ／ｘ＋ｙ×Ｙ／ｙとなる。

【０２０３】このような通信を、ｘ個のプロセッサエレメントがそれぞれ行うので、（ＲＴ）ペアにより指定される１つのジョブ当たりで、

ｘ×［Ｘ／ｘ＋ｙ×Ｙ／ｙ］＝Ｘ＋ｘＹ

の通信量となる。

【０２０４】上記の例とは逆に、（１，１）から（ｘ，１）をプロセッサエレメントＰＥ１に、（１，２）から（ｘ，２）をプロセッサエレメントＰＥ２に、（１，ｙ）から（ｘ，ｙ）をプロセッサエレメントＰＥｙに、というように、プロセッサエレメントへの割り付けを行う（第２の分割方法）場合、（ＲＴ）ペアにより指定される１つのジョブ当たりの通信量は、

ｙＸ＋Ｙ

となる。

【０２０５】プロセッサエレメント２内のメモリ容量は限られているので、縮約シェル番号の区切り数を少なくするためには元々のデータ量がなるべく小さい方の区切りで別々のプロセッサエレメント２にサブジョブを割り当てるのが良い。

【０２０６】第１の分割方法と第２の分割方法とで大きさを比較してみる。ｘはＸに比例し、ｙはＹに比例して、その比例定数は近似的に共通とすることができるので、第１の分割方法と第２の分割方法とでの通信量の差は、Ｘ－Ｙとなる。したがって通信量は、Ｘ＞Ｙの場合には、第２の分割方法の方が小さく、Ｘ＜Ｙの場合には、第１の分割方法の方が小さくなる。

【０２０７】すなわち、密度行列要素Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）とフォック行列要素Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の要素数の和が、密度行列要素Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）とフォック行列要素Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）の要素数の和よりも大きい（Ｘ＞Ｙ）場合には、第２の分割方法を用いる。逆に、密度行列要素Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）とフォック行列要素Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の要素数の和が、密度行列要素Ｐ（Ｉ，Ｌ），Ｐ（Ｋ，Ｌ）とフォック行列要素Ｆ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ）の要素数の和よりも小さい（Ｘ＜Ｙ）場合には、第１の分割方法を用いることにより、通信量の増大分を小さくすることが可能である。

【０２０８】図１４に、ジョブの分割を考慮した場合のホスト計算機における処理のアルゴリズムを示す。この図１４は、図５中のホスト計算機の処理アルゴリズムにおけるステップＳ１０３からステップＳ１０６までの部分に対応するもので、図１４のステップＳ４０１が図５のステップＳ１０３に、図１４のステップＳ４１１が図５のステップＳ１０６に、それぞれ対応している。なお、この図１４では、プロセッサエレメントは、簡単のためＰＥと記述した。

【０２０９】すなわち、図１４において、ステップＳ４０１では、プロセッサエレメント２に割り当てる（ＲＴ）ペア番号を決定する。その後、ホスト計算機１は、まず、（ＲＴ）ペア番号が割り当てられたプロセッサエレメント２のメモリに、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを格納可能かどうかを判断する（ステップＳ４０２）。

【０２１０】ステップＳ４０２で、格納可能であると判断されたときには、次に、（ＲＴ）ペア番号が割り当てられたプロセッサエレメント２のメモリに、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを格納可能かどうかを判断し（ステップＳ４０５）、格納可能であれば、ステップＳ４０６、ステップＳ４０７に順に進む。また、格納可能でなければ、後述する図１５に示す処理に進む。

【０２１１】ステップＳ４０６、４０７の処理は、プロセッサエレメント２に割り当てられた（ＲＴ）ペア番号に相当する処理に必要な密度行列およびフォック行列を、全て、プロセッサエレメント２のメモリに格納可能な場合の処理であるので、図５における処理と同様である。すなわち、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）の全てをプロセッサエレメント２へ送信する（ステップＳ４０６）。その後は、処理が終了して、プロセッサエレメント２から、Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを受信するまで、ホスト計算機は処理待ちの状態となる（ステップＳ４０７）。

【０２１２】また、ステップＳ４０２で、格納可能でないと判断されたときには、（ＲＴ）ペア番号が割り当てられたプロセッサエレメント２のメモリに、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てが格納可能かどうかを判断し（ステップＳ４０３）、格納可能であると判断されたときには、後述する図１６に示す処理を行い、そうでなければステップＳ４０４に進む。

【０２１３】ステップＳ４０４では、Ｐ（Ｉ，Ｊ）とＰ（Ｋ，Ｊ）の個数の和と、Ｐ（Ｋ，Ｌ）とＰ（Ｉ，Ｌ）の個数の和とを比較し、等しいかあるいは後者が大きければ、後述する図１７に示す処理を行ない、前者が大きければ、後述する図１８に示す処理を行なう。

【０２１４】図１５、図１６、図１７および図１８の処理のいずれかの処理が終了すると、ホスト計算機１は、全ての（ＲＴ）ペアの処理が終了したかどうかを判断する（ステップＳ４１１）。以下、図５の処理と同様である。

【０２１５】次に、図１５に示す処理を説明する。この処理は、プロセッサエレメント２に割り当てられた（ＲＴ）ペア番号に相当する処理に必要な密度行列およびフ

ォック行列の全ては、プロセッサエレメント２のメモリに格納できないが、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを格納することができる場合の処理である。

【０２１６】まず、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを格納した場合に、プロセッサエレメント２のメモリに残る空き容量を見積り、対応するＰ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てがその空き容量以下となるように、Ｓの範囲をＭ分割する。それとともに、分割された各Ｓの範囲を単位としてＭ個のサブジョブを定義する（ステップＳ４０８）。

【０２１７】次に、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ）の全てをプロセッサエレメント２に送信する（ステップＳ４０９）。その後、ｍ＝１からｍ＝Ｍまでに対して、ｍ番目のサブジョブに対応するＰ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）の全てをプロセッサエレメント２へ送信し、処理待ちの状態を経て、プロセッサエレメント２からｍ番目のサブジョブに対応するＦ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを受信する、という処理を繰り返す（ステップＳ４１０～ステップＳ４１４）。

【０２１８】そして、ｍ＝１からｍ＝Ｍまでの繰り返しの処理の全てを終了すると、プロセッサエレメントからＦ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを受信する（ステップＳ４１５）。その後、図１４のステップＳ４１１に進む。

【０２１９】次に、図１６に示す処理を説明する。これは、プロセッサエレメント２に割り当てられた（ＲＴ）ペア番号に相当する処理に必要な密度行列およびフォック行列の全てをプロセッサエレメント２のメモリに格納できず、また、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを格納することもできないが、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを格納することができる場合の処理である。

【０２２０】まず、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを格納した場合に、プロセッサエレメント２のメモリに残る空き容量を見積り、対応するＰ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てががその空き容量以下となるようにＵの範囲をＭ分割する。それとともに、分割された各Ｕの範囲を単位としてＭ個のサブジョブを定義する（ステップＳ４１６）。

【０２２１】次に、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）の全てをプロセッサエレメント２に送信する（ステップＳ４１７）。その後、ｍ＝１からｍ＝Ｍまでに対して、ｍ番目のサブジョブに対応するＰ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ）の全てをプロセッサエレメントへ送信し、処理待ちの状態を経て、プロセッサエレメントからｍ番目のサブジョブに対応するＦ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを受信する、という処理を繰り返す（ステップＳ４１８～ステップＳ４２２）。

【０２２２】そして、ｍ＝１からｍ＝Ｍまでの繰り返しの処理の全てを終了すると、プロセッサエレメント２からＦ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを受信する（ステップＳ４２３）。その後、図１４のステップＳ４１１に進む。

【０２２３】次に、図１７に示す処理を説明する。これは、プロセッサエレメント２のメモリに、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを格納することができず、また、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを格納することもできない場合で、Ｐ（Ｉ，Ｊ）とＰ（Ｋ，Ｊ）の個数の和がＰ（Ｋ，Ｌ）のＰ（Ｉ，Ｌ）個数の和よりも大きくない場合の処理である。

【０２２４】まず、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てがメモリの空き容量以下となるように、Ｓの範囲をＭ１分割する。Ｍ１はできるだけ小さく設定する（ステップＳ４２４）。

【０２２５】次に、ステップＳ４２５で、ｍ１＝０として初期化した後、ステップＳ４２６以降に進み、ｍ１に関するループを、ｍ１＝１からｍ１＝Ｍ１の範囲で回す。ｍ１に関するループの中では、まず、分割されたＰ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを格納した場合に、プロセッサエレメント２のメモリに残る空き容量を見積り、対応するＰ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てがその空き容量以下となるようにＵの範囲をＭ２分割する。さらに、分割された各Ｕの範囲を単位としてＭ２個のサブジョブを定義する（ステップＳ４２７）。そして、次に、ｍ１番目の分割に対応するＰ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）の全てを、プロセッサエレメント２へ送信する（ステップＳ４２８）。

【０２２６】その後、ステップＳ４２９で、ｍ２＝０として初期化した後、ステップＳ４３０以降に進み、ｍ２に関するループを、ｍ２＝１からｍ２＝Ｍ２の範囲で回す。ｍ２に関するループでは、まず、ｍ２番目の分割に対応するＰ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ）の全てをプロセッサエレメントへ送信し（ステップＳ４３１）、処理待ちの状態を経て、プロセッサエレメントからｍ２番目の分割に対応するＦ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを受信する（ステップＳ４３２）。

【０２２７】ｍ２に関するループが終了したと判断すると（ステップＳ４３３）、プロセッサエレメント２からｍ１番目の分割に対応するＦ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを受信する（ステップＳ４３４）。そして、ｍ１に関するループが終了したと判断すると（ステップＳ４３５）、図１４のステップＳ４１１に進む。

【０２２８】なお、ｍ１番目の分割に対応するループ内の全てのサブジョブの処理は、同一のプロセッサエレメ

ント２で行なう必要があるが、Ｍ１個に分割された異なるＳの範囲に対応する処理は、それぞれを独立のジョブと見なして、別々のプロセッサエレメントで処理を行なってもよい。それらのジョブを、同一のプロセッサエレメントで行なっても、別々のプロセッサエレメントで行なっても、発生する通信量は同じである。

【０２２９】次に、図１８に示す処理を説明する。この処理は、プロセッサエレメント２のメモリに、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを格納することができず、また、Ｐ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを格納することもできない場合で、Ｐ（Ｉ，Ｊ）とＰ（Ｋ，Ｊ）の個数の和が、Ｐ（Ｋ，Ｌ）のＰ（Ｉ，Ｌ）個数の和よりも大きい場合の処理である。

【０２３０】まず、Ｐ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てがメモリの空き容量以下となるように、Ｕの範囲をＭ１分割する。Ｍ１はできるだけ小さく設定する（ステップＳ４３６）。次に、ステップＳ４３７で、ｍ１＝０として初期化した後、ステップＳ４３８以降に進み、ｍ１に関するループを、ｍ１＝１からｍ１＝Ｍ１の範囲で回す。

【０２３１】ｍ１に関するループの中では、まず、分割されたＰ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ），Ｆ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを格納した場合に、プロセッサエレメントのメモリに残る空き容量を見積り、対応するＰ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ），Ｆ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てがその空き容量以下となるように、Ｓの範囲をＭ２分割する。さらに、分割された各Ｓの範囲を単位としてＭ２個のサブジョブを定義する（ステップＳ４３９）。

【０２３２】ｍ１番目の分割に対応するＰ（Ｋ，Ｌ），Ｐ（Ｉ，Ｌ）の全てをプロセッサエレメント２へ送信した後、ステップＳ４４１で、ｍ２＝０として初期化した後、ステップＳ４４２以降に進み、ｍ２に関するループを、ｍ２＝１からｍ２＝Ｍ２の範囲で回す。

【０２３３】ｍ２に関するループでは、まず、ｍ２番目の分割に対応するＰ（Ｉ，Ｊ），Ｐ（Ｋ，Ｊ）の全てをプロセッサエレメント２へ送信し（ステップＳ４４３）、処理待ちの状態を経て、プロセッサエレメント２からｍ２番目の分割に対応するＦ（Ｉ，Ｊ），Ｆ（Ｋ，Ｊ）の全てを受信する（ステップＳ４４４）。

【０２３４】ｍ２に関するループが終了したと判断すると（ステップＳ４４５）、プロセッサエレメント２からｍ１番目の分割に対応するＦ（Ｋ，Ｌ），Ｆ（Ｉ，Ｌ）の全てを受信する（ステップＳ４４６）。そして、ｍ１に関するループが終了したと判断すると（ステップＳ４４７）、図１４のステップＳ４１１に進む。

【０２３５】なお、この場合も、ｍ１番目の分割に対応するループ内の全てのサブジョブの処理は、同一のプロセッサエレメント２で行なう必要があるが、Ｍ１個に分割された異なるＵの範囲に対応する処理は、それぞれを独立のジョブと見なして、別々のプロセッサエレメント２で処理を行なってもよい。

【０２３６】以上説明した実施の形態は、非経験的分子軌道法を用いた分子シミュレーションにおいて、フォック行列要素計算を高速に行う場合に、この発明を適用した場合であるが、この発明は、このような非経験的分子軌道法に限らず、種々の並列処理アルゴリズムに適用可能であることは、言うまでもない。

【０２３７】

【発明の効果】以上説明したように、この発明によれば、安価な通信手段と比較的小さなメモリを持った並列処理システムを用いて、大規模な行列要素の計算を高速に行うことが可能となる。

【図面の簡単な説明】

【図１】この発明による並列処理装置の実施の形態のシステム構成を示すブロック図である。

【図２】この発明の実施の形態の比較例として示す従来例のＦｏｓｔｅｒの列ブロックアルゴリズムでトリプルソート法のアルゴリズムを示すプログラム・コードを示す図である。

【図３】この発明の実施の形態のＲＴ並列アルゴリズムを示すプログラム・コードを示す図である。

【図４】この発明の実施の形態のＲＴ並列アルゴリズムにおいて、必要な通信性能と縮約シェル番号Ｒとの関係を示す図である。

【図５】この発明の実施の形態におけるホスト計算機およびプロセッサエレメントの処理のフローチャートである。

【図６】この発明の実施の形態におけるホスト計算機上に用意されるカットオフテーブルの一例を示す図である。

【図７】この発明の実施の形態において、ホスト計算機からプロセッサエレメントへ送信される密度行列情報のフォーマットの一例を示す図である。

【図８】この発明の実施の形態における密度行列データブロックの構成例を示す図である。

【図９】この発明の実施の形態において、プロセッサエレメントからホスト計算機へ送信されるフォック行列情報のフォーマットの一例を示す図である。

【図１０】この発明の実施の形態におけるメモリへの行列情報の割り付け例を示す図である。

【図１１】この発明の実施の形態におけるプロセッサエレメントの処理のフローチャートの一部を示す図である。

【図１２】この発明の実施の形態におけるプロセッサエレメントの処理のフローチャートの一部を示す図である。

【図１３】この発明の実施の形態における、計算規模が大きい場合のサブジョブへの分割と、それらのプロセッ

サエレメントへの割り当て方法の一例を示す図である。

【図１４】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図１５】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図１６】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図１７】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図１８】この発明の実施の形態における、ジョブ分割を考慮した場合のホスト計算機の処理アルゴリズムの一例を示すフローチャートの一部を示す図である。

【図１９】原始基底関数と、その角運動量、軌道指数、原子核座標との対応例を示す図である。

【図２０】従来例のＦｏｓｔｅｒの列ブロックアルゴリズムのフローチャートである。

【図２１】従来例のＦｏｓｔｅｒの列ブロックアルゴリズムでカノニカル法の場合における、必要な通信性能のブロックサイズ依存性を説明するための図である。

【図２２】従来例のＦｏｓｔｅｒの列ブロックアルゴリズムでトリプルソート法の場合における、必要な通信性能のブロックサイズ依存性を説明するための図である。

【図２３】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図２４】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図２５】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図２６】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図２７】非経験的分子軌道計算法の説明に用いる数式を示す図である。

【図２８】文献１の２電子積分計算手法の説明に用いる数式を示す図である。

【図２９】文献１の２電子積分計算手法の説明に用いる数式を示す図である。

【図３０】従来例のＦｏｓｔｅｒのアルゴリズムの説明に用いる数式を示す図である。

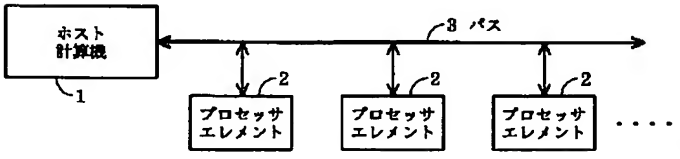
【図３１】従来例のＦｏｓｔｅｒのアルゴリズムの説明に用いる数式を示す図である。

【図３２】従来例における通信量（数式４４）、従来例における必要な通信性能（数式４５）、この発明の実施の形態における必要な通信性能（数式４６）、この発明の実施の形態におけるカットオフの判定基準の一例（数式４７）、この発明の実施の形態におけるＲＴペア番号の一意的な決め方の一例（数式４８）に、それぞれ用いる数式を示す図である。

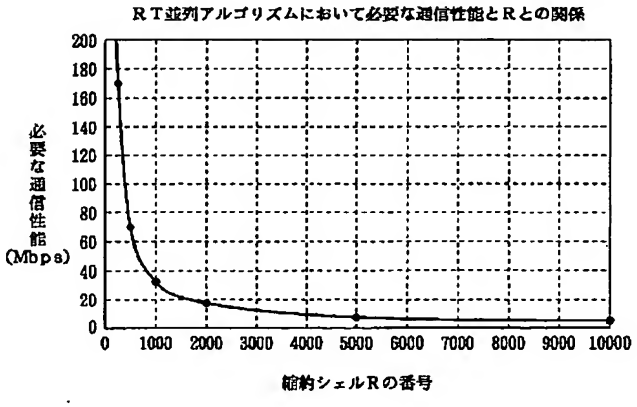
【符号の説明】

1　ホスト計算機
2　プロセッサエレメント
3　バス

【図１】



【図４】



【図６】

カットオフテーブルの一例

縮約シェルＸの番号	縮約シェルＸとのペアでのカットオフで生き残る縮約シェル番号
1	1, 2, 3, 5
2	2, 4
3	1, 3, 5,
4	2, 3, 4, 6
5	1, 3, 5, 6, 7
⋮	⋮
100	87, 91, 92, 96, 97, 98, 99, 100

【図２】

トリプルソートアルゴリズム（比較例）

```
for(R=1;R<=Nshell;R++){
for(S=1;S<=R;S++){
for(T=1;T<=S;T++){
for(U=1;U<=T;U++){
  for(I=b_basis(R);I<=e_basis(R);I++){
  for(J=b_basis(S);J<=e_basis(S);J++){
  for(K=b_basis(T);K<=e_basis(T);K++){
  for(L=b_basis(U);L<=e_basis(U);L++){
    G_IJKL=G(I,J,K,L);
    G_IKJL=G(I,K,J,L);
    G_ILJK=G(I,L,J,K);
    F[I][J]+=P[K][L]*(G_IJKL+G_IKJL+G_ILJK);
    F[I][K]-=P[J][L]*(G_IJKL+G_IKJL+G_ILJK)/2;
    F[I][L]-=P[J][K]*(G_IJKL+G_IKJL+G_ILJK)/2;
    F[J][K]-=P[I][L]*(G_IJKL+G_IKJL+G_ILJK)/2;
    F[J][L]-=P[I][K]*(G_IJKL+G_IKJL+G_ILJK)/2;
    F[K][L]+=P[I][J]*(G_IJKL+G_IKJL+G_ILJK);
  }}}}
}}}}
```

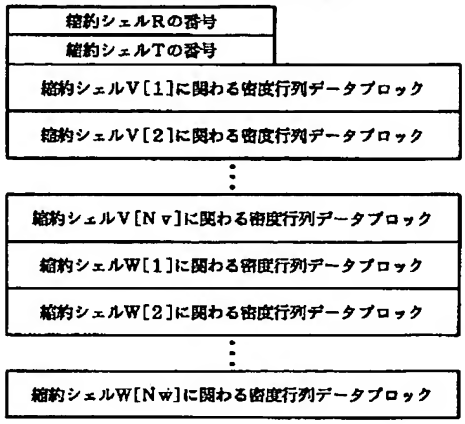
【図３】

RT並列アルゴリズム（この発明の実施形態）

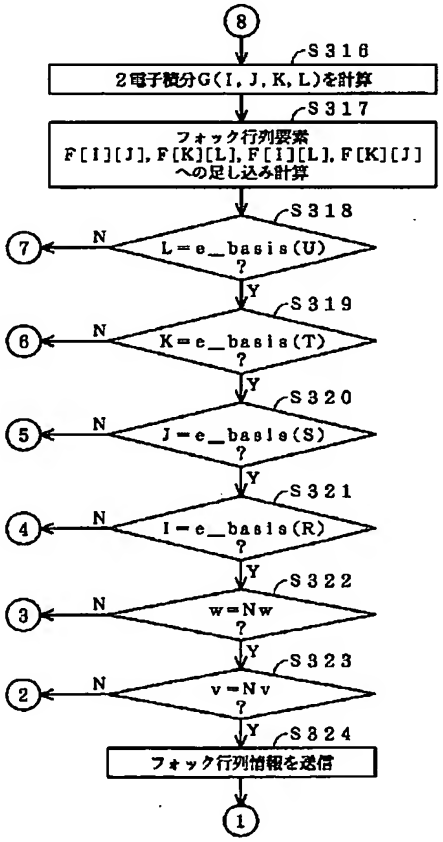
```
for(RT=1;RT<=Nshell*(Nshell+1)/2;RT++){
              for(S=1;S<=R;S++){
              for(U=1;U<=R;U++){
                for(I=b_basis(R);I<=e_basis(R);I++){
                for(J=b_basis(S);J<=e_basis(S);J++){
                for(K=b_basis(T);K<=e_basis(T);K++){
                for(L=b_basis(U);L<=e_basis(U);L++){
                  G_IJKL=G(I,J,K,L);
                  F[I][J]+=P[K][L]*G_IJKL;
                  if(L<=K&&K<I){
                    if(J<I)  F[K][L]+=2*P[I][J]*G_IJKL;
                    else     F[K][L]+=P[I][J]*G_IJKL;
                  }
                  F[I][L]-=0.5*P[K][J]*G_IJKL;
                  if(J<I&&K<=J)
                            F[K][J]-=0.5*P[I][L]*G_IJKL;
                  if(K<I&&J<=K&&L<I)
                            F[K][J]-=0.5*P[I][L]*G_IJKL;
                }}}}
              }}
}}
```

【図９】

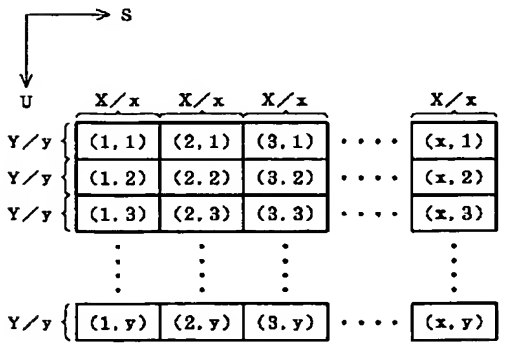
転送するフォック行列情報のフォーマット例



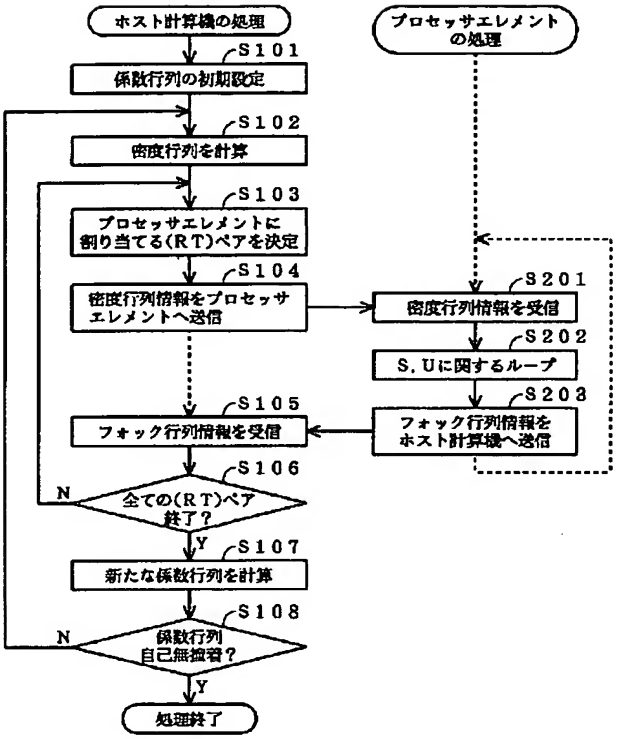
【図１２】



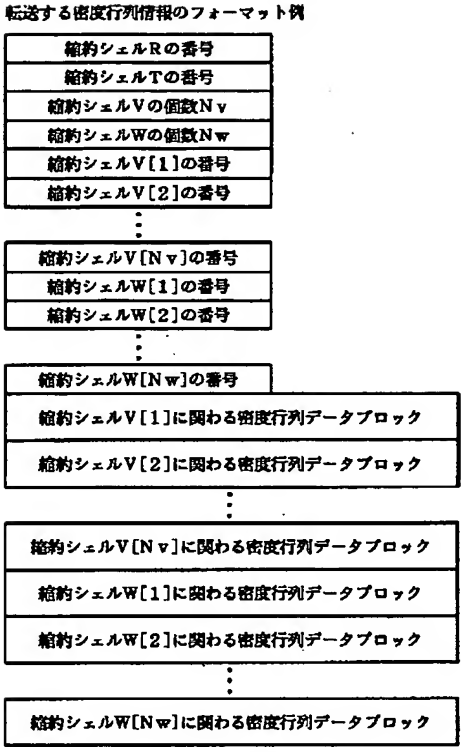
【図１３】



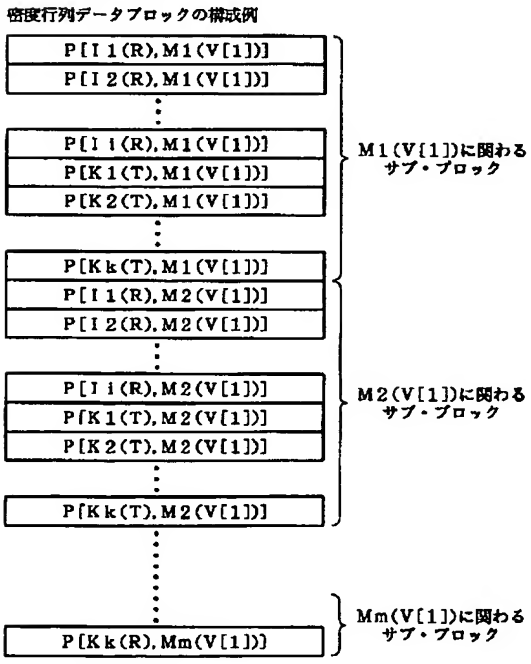
【図５】


ホスト計算機の処理
プロセッサエレメントの処理
S101
係数行列の初期設定
S102
密度行列を計算
S103
プロセッサエレメントに割り当てる(RT)ペアを決定
S104
密度行列情報をプロセッサエレメントへ送信
S201
密度行列情報を受信
S202
S,Uに関するループ
S203
フォック行列情報をホスト計算機へ送信
S105
フォック行列情報を受信
S106
全ての(RT)ペア終了?
N
Y
S107
新たな係数行列を計算
S108
係数行列自己無撞着?
N
Y
処理終了

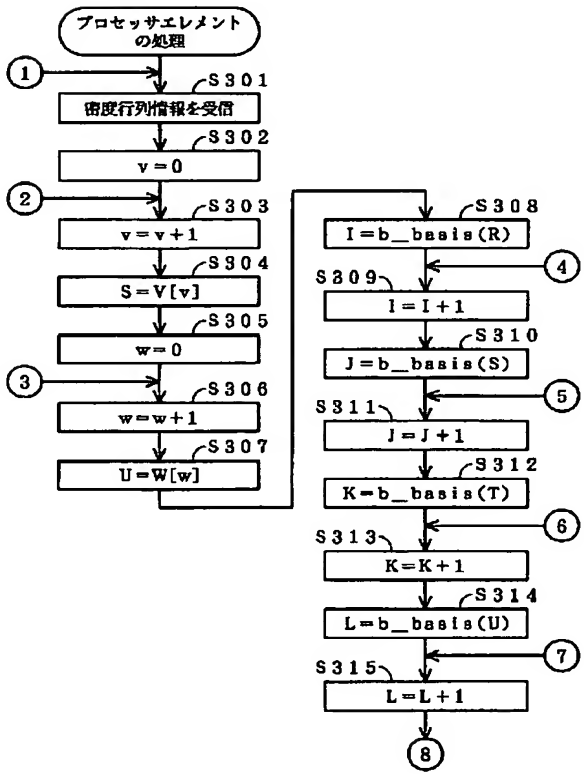
【図７】


転送する密度行列情報のフォーマット例
縮約シェルRの番号
縮約シェルTの番号
縮約シェルVの個数Nv
縮約シェルWの個数Nw
縮約シェルV[1]の番号
縮約シェルV[2]の番号
縮約シェルV[Nv]の番号
縮約シェルW[1]の番号
縮約シェルW[2]の番号
縮約シェルW[Nw]の番号
縮約シェルV[1]に関わる密度行列データブロック
縮約シェルV[2]に関わる密度行列データブロック
縮約シェルV[Nv]に関わる密度行列データブロック
縮約シェルW[1]に関わる密度行列データブロック
縮約シェルW[2]に関わる密度行列データブロック
縮約シェルW[Nw]に関わる密度行列データブロック

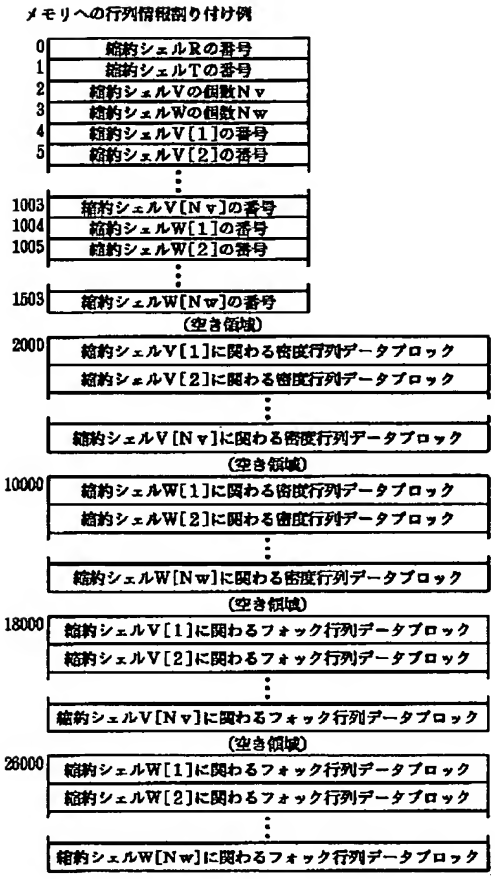
【図８】


密度行列データブロックの構成例
P[I1(R),M1(V[1])]
P[I2(R),M1(V[1])]
P[Ii(R),M1(V[1])]
P[K1(T),M1(V[1])]
P[K2(T),M1(V[1])]
P[Kk(T),M1(V[1])]
P[I1(R),M2(V[1])]
P[I2(R),M2(V[1])]
P[Ii(R),M2(V[1])]
P[K1(T),M2(V[1])]
P[K2(T),M2(V[1])]
P[Kk(T),M2(V[1])]
P[Kk(R),Mm(V[1])]
M1(V[1])に関わるサブ・ブロック
M2(V[1])に関わるサブ・ブロック
Mm(V[1])に関わるサブ・ブロック

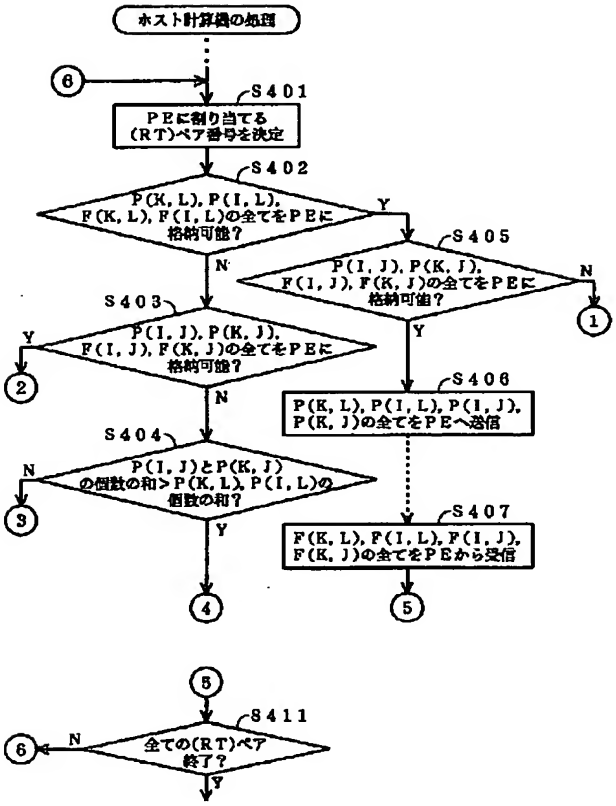
【図１１】


プロセッサエレメントの処理
1
S301
密度行列情報を受信
S302
v=0
2
S303
v=v+1
S304
S=V[v]
S305
w=0
3
S306
w=w+1
S307
U=W[w]
S308
I=b_basis(R)
4
S309
I=I+1
S310
J=b_basis(S)
5
S311
J=J+1
S312
K=b_basis(T)
6
S313
K=K+1
S314
L=b_basis(U)
7
S315
L=L+1
8

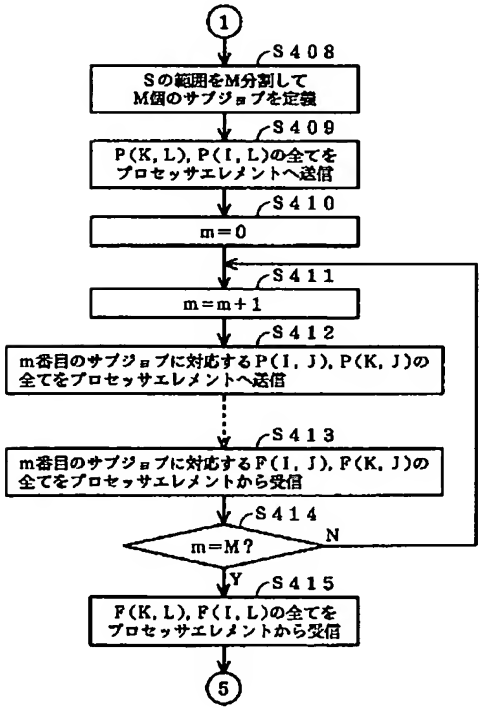
【図１０】


メモリへの行列情報割り付け例
0 縮約シェルRの番号
1 縮約シェルTの番号
2 縮約シェルVの個数Nv
3 縮約シェルWの個数Nw
4 縮約シェルV[1]の番号
5 縮約シェルV[2]の番号
1003 縮約シェルV[Nv]の番号
1004 縮約シェルW[1]の番号
1005 縮約シェルW[2]の番号
1503 縮約シェルW[Nw]の番号
(空き領域)
2000 縮約シェルV[1]に関わる密度行列データブロック
縮約シェルV[2]に関わる密度行列データブロック
縮約シェルV[Nv]に関わる密度行列データブロック
(空き領域)
10000 縮約シェルW[1]に関わる密度行列データブロック
縮約シェルW[2]に関わる密度行列データブロック
縮約シェルW[Nw]に関わる密度行列データブロック
(空き領域)
18000 縮約シェルV[1]に関わるフォック行列データブロック
縮約シェルV[2]に関わるフォック行列データブロック
縮約シェルV[Nv]に関わるフォック行列データブロック
(空き領域)
26000 縮約シェルW[1]に関わるフォック行列データブロック
縮約シェルW[2]に関わるフォック行列データブロック
縮約シェルW[Nw]に関わるフォック行列データブロック

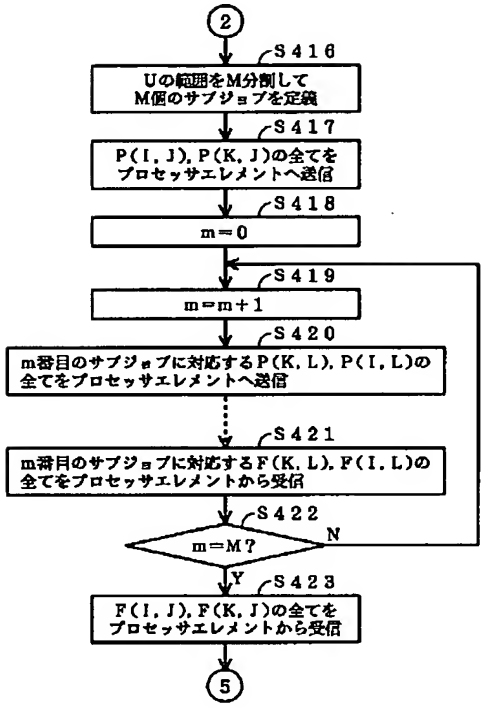
【図１４】


ホスト計算機の処理
S401
PEに割り当てる(RT)ペア番号を決定
S402
P(K, L), P(I, L), F(K, L), F(I, L)の全てをPEに格納可能?
Y
N
S405
P(I, J), P(K, J), F(I, J), F(K, J)の全てをPEに格納可能?
N
Y
1
S403
P(I, J), P(K, J), F(I, J), F(K, J)の全てをPEに格納可能?
Y
2
N
S406
P(K, L), P(I, L), P(I, J), P(K, J)の全てをPEへ送信
S404
P(I, J)とP(K, J)の個数の和>P(K, L), P(I, L)の個数の和?
N
3
Y
S407
F(K, L), F(I, L), F(I, J), F(K, J)の全てをPEから受信
4
5
5
S411
6
N
全ての(RT)ペア終了?
Y

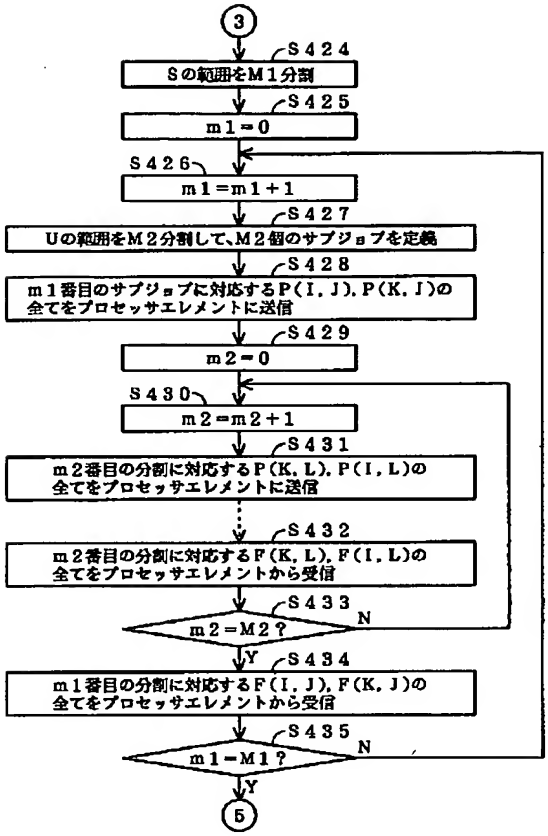
【図１５】


1
S408
Sの範囲をM分割してM個のサブジョブを定義
S409
P(K, L), P(I, L)の全てをプロセッサエレメントへ送信
S410
m=0
S411
m=m+1
S412
m番目のサブジョブに対応するP(I, J), P(K, J)の全てをプロセッサエレメントへ送信
S413
m番目のサブジョブに対応するF(I, J), F(K, J)の全てをプロセッサエレメントから受信
S414
m=M?
N
Y
S415
F(K, L), F(I, L)の全てをプロセッサエレメントから受信
5

【図１６】



【図１７】

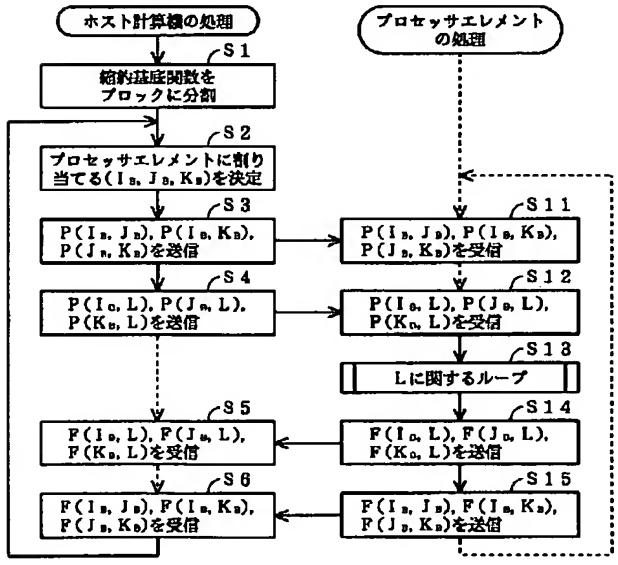


【図１９】

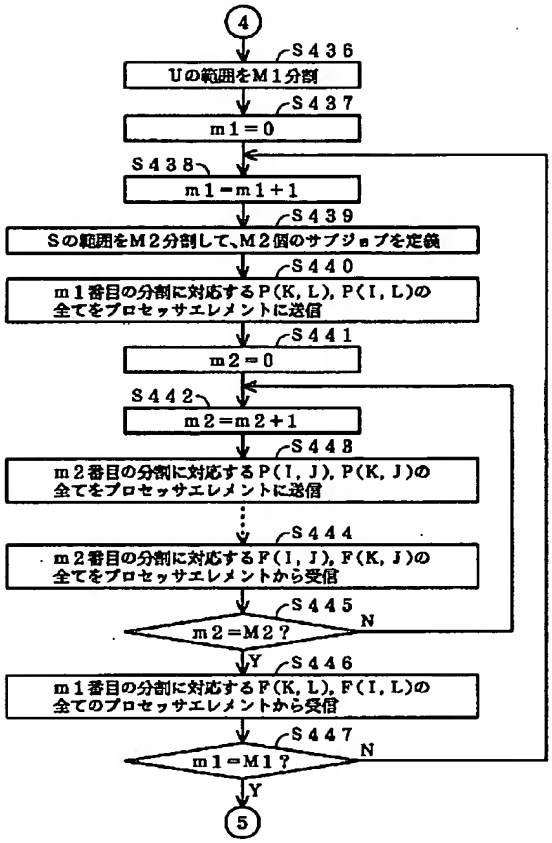
[表1　原始基底関数i, j, k, lの角運動量、軌道指数、原子核座標]

原始基底関数の番号	角運動量	軌道指数	原子核座標
i	a	ζ_a	A
j	b	ζ_b	B
k	c	ζ_c	C
l	d	ζ_d	D

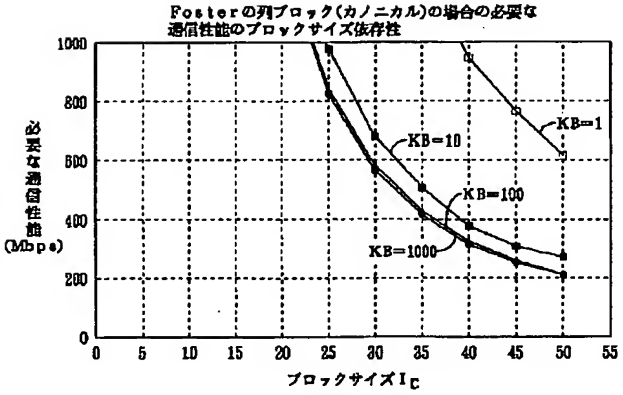
【図２０】



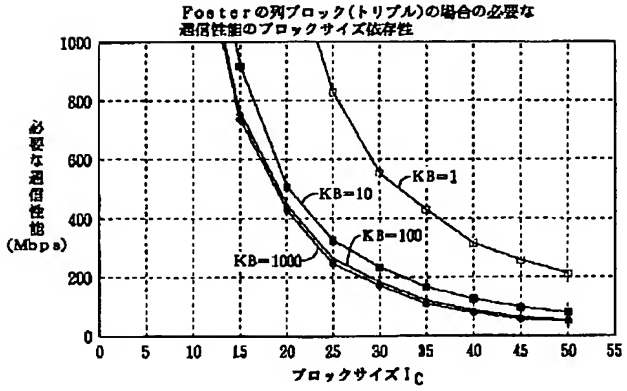
【図１８】



【図２１】



【図２２】



【図２３】

(数式1)

$$\phi_\mu = \sum_I \chi_I C_{I\mu}$$

(数式2)

$$\Psi(1,2,\cdots,2n) = \frac{1}{\sqrt{(2n)!}} \times \begin{vmatrix} \phi_1\alpha(1) & \phi_1\beta(1) & \phi_2\alpha(1) & \phi_2\beta(1) & \cdots & \phi_n\alpha(1) & \phi_n\beta(1) \\ \phi_1\alpha(2) & \phi_1\beta(2) & \phi_2\alpha(2) & \phi_2\beta(2) & \cdots & \phi_n\alpha(2) & \phi_n\beta(2) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \phi_1\alpha(2n) & \phi_1\beta(2n) & \phi_2\alpha(2n) & \phi_2\beta(2n) & \cdots & \phi_n\alpha(2n) & \phi_n\beta(2n) \end{vmatrix}$$

(数式3)

$$H = H_1 + H_2$$

(数式4)

$$H_1 = \sum_p \left[-\frac{1}{2}\nabla_p^2 - \sum_A \frac{Z_A}{r_{pA}} \right]$$

(数式5)

$$H_2 = \sum_p \sum_{q(>p)} \frac{1}{r_{pq}}$$

【図２４】

(数式6)

$$\varepsilon = \int \Psi H \Psi \, d\tau = \int \Psi (H_1 + H_2) \Psi \, d\tau$$
$$= 2 \sum_{\mu} H_{\mu} + \sum_{\mu} \sum_{\nu} (2 J_{\mu\nu} - K_{\mu\nu})$$

(数式7)

$$H_{\mu} = \int \phi_{\mu}(1) h^{core}(1) \phi_{\mu}(1) \, d\tau_1$$

(数式8)

$$J_{\mu\nu} = \int\int \phi_{\mu}(1) \phi_{\nu}(2) \frac{1}{r_{12}} \phi_{\mu}(1) \phi_{\nu}(2) \, d\tau_1 d\tau_2$$

(数式9)

$$K_{\mu\nu} = \int\int \phi_{\mu}(1) \phi_{\nu}(2) \frac{1}{r_{12}} \phi_{\mu}(2) \phi_{\nu}(1) \, d\tau_1 d\tau_2$$

【図２７】

(数式20)

$$\chi(r, n, R) = (r_x - R_x)^{nx} (r_y - R_y)^{ny} (r_z - R_z)^{nz} \times \sum_{m} d_m \exp\left[-\zeta_m (r-R)^2\right]$$

(数式21)

$$\phi(r, n, R) = (r_x - R_x)^{nx} (r_y - R_y)^{ny} (r_z - R_z)^{nz} \exp\left[-\zeta (r-R)^2\right]$$

(数式22)

$$\sum_{m} d_m \exp\left[-\zeta_m (r-R)^2\right]$$

(数式23)

$$G(I, J, K, L) = \sum_{m1} \sum_{m2} \sum_{m3} \sum_{m4} d_{m1} d_{m2} d_{m3} d_{m4} g(i, j, k, l)$$

(数式24)

$$g(i, j, k, l) = \int\int \phi_i(1) \phi_j(1) \frac{1}{r_{12}} \phi_k(2) \phi_l(2) \, d\tau_1 d\tau_2$$

(数式25)

$$g(i, j, k, l) = g(i, j, l, k) = g(j, i, l, k) = g(j, i, l, k) = g(k, l, i, j) = g(l, k, i, j) = g(k, l, j, i) = g(l, k, j, i)$$

【図２５】

(数式１０)

$$e=2\sum_{\mu}\left[\sum_{I}\sum_{J}C_{I\mu}C_{J\nu}H_{IJ}\right]+\sum_{\mu}\sum_{\nu}\left\{\sum_{I}\sum_{J}\sum_{K}\sum_{L}C_{I\mu}C_{J\mu}C_{K\nu}C_{L\nu}[2G(I,J,K,L)-G(I,K,J,L)]\right\}$$

(数式１１)

$$H_{IJ}=\int \chi_I(1)\,h^{core}(1)\,\chi_J(1)\,d\tau_1$$

(数式１２)

$$h^{core}(1)=-\frac{1}{2}\nabla_1^2-\sum_{A}\frac{Z_A}{r_{1A}}$$

(数式１３)

$$G(I,J,K,L)=\int\int \chi_I(1)\,\chi_J(1)\frac{1}{r_{12}}\chi_K(2)\,\chi_L(2)\,d\tau_1 d\tau_2$$

【図２９】

(数式３５)

$$\begin{aligned}&[(a+1_i)\,b,\,c\,d]^{(m)}\\&=(P_i-A_i)\,[a\,b,\,c\,d]^{(m)}+(W_i-P_i)\,[a\,b,\,c\,d]^{(m+1)}\\&+\frac{N_i(a)}{2\zeta}\left\{[(a-1_i)\,b,\,c\,d]^{(m)}-\frac{\rho}{\zeta}[(a-1_i)\,b,\,c\,d]^{(m+1)}\right\}\\&+\frac{N_i(b)}{2\zeta}\left\{[a\,(b-1_i),\,c\,d]^{(m)}-\frac{\rho}{\zeta}[a\,(b-1_i),\,c\,d]^{(m+1)}\right\}\\&+\frac{N_i(c)}{2(\zeta+\eta)}[a\,b,\,(c-1_i)\,d]^{(m+1)}\\&+\frac{N_i(d)}{2(\zeta+\eta)}[a\,b,\,c\,(d-1_i)]^{(m+1)}\end{aligned}$$

(数式３６)

$$W=\frac{\zeta P+\eta Q}{\zeta+\eta}$$

(数式３７)

$$exp=\left[-\frac{\zeta\zeta'(R-R')^2}{\zeta+\zeta'}\right]$$

【図２６】

(数式１４)

$$\sum_{J}(F_{IJ}-\varepsilon_{\mu}S_{IJ})C_{J\mu}=0$$

(数式１５)

$$F_{IJ}=H_{IJ}+\sum_{K}\sum_{L}P_{KL}\left[G(I,J,K,L)-\frac{1}{2}G(I,K,J,L)\right]$$

(数式１６)

$$\varepsilon_{\mu}=H_{\mu}+\sum_{\nu}(2J_{\mu\nu}-K_{\mu\nu})$$

(数式１７)

$$S_{IJ}=\int\chi_{I}(1)\chi_{J}(1)d\tau_{1}$$

(数式１８)

$$P_{KL}=2\sum_{\nu}C_{K\nu}C_{L\nu}$$

(数式１９)

$$G(I,J,K,L)=G(I,J,L,K)=G(J,I,K,L)=G(J,I,L,K)=$$
$$G(K,L,I,J)=G(L,K,I,J)=G(K,L,J,I)=G(L,K,J,I)$$

【図３０】

(数式３８)

$$I\geq J,K\geq L,(IJ)\geq(KL)$$

$$(IJ)=\frac{I(I-1)}{2}+J,(KL)=\frac{K(K-1)}{2}+L$$

(数式３９)

$$F_{IJ}+=P_{KL}G(I,J,K,L),$$
$$F_{IK}+=-\frac{1}{2}P_{JL}G(I,J,K,L),$$
$$F_{IL}+=-\frac{1}{2}P_{JK}G(I,J,K,L),$$
$$F_{JK}+=-\frac{1}{2}P_{IL}G(I,J,K,L),$$
$$F_{JL}+=-\frac{1}{2}P_{IK}G(I,J,K,L),$$
$$F_{KL}+=P_{IJ}G(I,J,K,L)$$

【図２８】

(数式２６)

$$[0_a 0_b, 0_c 0_d]^{(m)} = \frac{1}{\sqrt{\zeta+\eta}} \cdot K(\zeta_a, \zeta_b, A, B) \cdot K(\zeta_c, \zeta_d, C, D) \cdot F_m(T)$$

(数式２７)

$$F_m(T) = \int t^{2m} \exp(-T t^2) dt$$

(数式２８)

$$T = \rho (P-Q)^2$$

(数式２９)

$$K(\zeta, \zeta', R, R') = \sqrt{2} \frac{\pi^{5/4}}{\zeta+\zeta'} \exp\left[-\frac{\zeta \zeta' (R-R')^2}{\zeta+\zeta'}\right]$$

(数式３０)

$$\zeta = \zeta_a + \zeta_b$$

(数式３１)

$$\eta = \zeta_c + \zeta_d$$

(数式３２)

$$\rho = \frac{\zeta \eta}{\zeta+\eta}$$

(数式３３)

$$P = \frac{\zeta_a A + \zeta_b B}{\zeta_a + \zeta_b}$$

(数式３４)

$$Q = \frac{\zeta_c C + \zeta_d D}{\zeta_c + \zeta_d}$$

【図３１】

(数式４０)

$$I \geq J \geq K \geq L$$

(数式４１)

$$F_{IJ} += P_{KL} [G(I,J,K,L)+G(I,K,J,L)+G(I,L,J,K)],$$
$$F_{IK} -= P_{JL} [G(I,J,K,L)+G(I,K,J,L)+G(I,L,J,K)] / 2,$$
$$F_{IL} -= P_{JK} [G(I,J,K,L)+G(I,K,J,L)+G(I,L,J,K)] / 2,$$
$$F_{JK} -= P_{IL} [G(I,J,K,L)+G(I,K,J,L)+G(I,L,J,K)] / 2,$$
$$F_{JL} -= P_{IK} [G(I,J,K,L)+G(I,K,J,L)+G(I,L,J,K)] / 2,$$
$$F_{KL} += P_{IJ} [G(I,J,K,L)+G(I,K,J,L)+G(I,L,J,K)]$$

(数式４２)

$$I_B \geq J_B \geq K_B \geq L_B$$

(数式４３)

$$I_B \geq J_B, K_B \geq L_B, (I_B J_B) \geq (K_B L_B)$$

【図３２】

(数式４４)

$$\frac{2(2I_C^2+K_BI_C^2)\times64\times M}{T_{eri}\times\alpha^2K_BI_C^4}=\frac{(2+K_B)\times128M}{T_{eri}\alpha^2K_BI_C^2}\quad(Mbps)$$

(数式４５)

$$\frac{2(2I_C^2+K_BI_C^2)\times64\times M}{T_{eri}\times3\alpha^2K_BI_C^4}=\frac{(2+K_B)\times128M}{3T_{eri}\alpha^2K_BI_C^2}\quad(Mbps)$$

(数式４６)

$$\frac{8\alpha_s^3R\times64\times M}{T_{eri}\times\alpha^2\alpha_s^4R^2}=\frac{512M}{T_{eri}\alpha_sR}\quad(Mbps)$$

(数式４７)

$$\exp\left[-\frac{\zeta_A\zeta_B(A-B)^2}{\zeta_A+\zeta_B}\right]<10^{-15}$$

(数式４８)

$$(RT)=\frac{R(R-1)}{2}+T$$

フロントページの続き

(72)発明者　稲畑　深二郎
神奈川県足柄上郡中井町境430　グリーンテクなかい富士ゼロックス株式会社内
(72)発明者　宮川　宣明
神奈川県足柄上郡中井町境430　グリーンテクなかい富士ゼロックス株式会社内
(72)発明者　高島　一
東京都豊島区高田３－24－１　大正製薬株式会社内
(72)発明者　北村　一泰
東京都豊島区高田３－24－１　大正製薬株式会社内
(72)発明者　小原　繁
北海道釧路市武佐３－12－358
Ｆターム(参考)　5B045 AA07 GG12 KK04
5B049 BB07 EE03 EE04 EE41 GG07